UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA EN INFORMÁTICA

PROYECTO FIN DE CARRERA

Mejora de la herramienta MindReader. Adquisición y análisis de señales EEG.

Autor:
María Teresa Luque Ibañes
Tutores:
Ricardo Aler Mur
Inés María Galván León

8 de junio de 2010

Agradecimientos

Este proyecto nunca se habría llevado a cabo sin la ayuda de mis tutores, Ricardo Aler Mur e Inés María Galván León. Quiero agradecerles la confianza depositada en mi, sus consejos, su paciencia, y su dedicación.

Los tres voluntarios que participaron durante las tediosas sesiones de adquisición de datos, David, Pablo y Marcos, merecen un "gracias monumental" (David tres). Porque casi no se quejaron, por poner su tiempo y cuero cabelludo a mi disposición, por el interés mostrado en el proyecto antes, durante y después de las sesiones y por sus ánimos.

También quiero agradecer a mi cuñada Pilar García García (Doctora en Farmacia) la ayuda prestada a la hora de buscar artículos científicos sobre BCI, y a Juan Manuel Rodríguez Bernabeu (médico estomatólogo) por sus aclaraciones sobre conceptos básicos de neurología. A Edulis por sus lecciones magistrales sobre cómo aumentar la calidad de gráficos e imágenes. A Rosa, Patricia y Alberto por sus aclaraciones sobre el uso del procesador de textos LaTeX.

A mis amiguitos de la universidad, con quienes he compartido clases, sabores, sinsabores y *arturocantoblanco*, les agradezco el apoyo que me han dado en los asuntos de clase y en los asuntos de la vida. David, Isra, Antonio, Víctor, Raquel, Bea, Elena y Sara han sido una gran motivación para terminar este PFC, aunque sólo sea por tener una excusa para reunirnos todos a celebrarlo. Además, algunos de ellos me han dado consejos valiosísimos para la realización de esta memoria.

Otros que me han acompañado y apoyado tanto en la cotidianidad como en momentos críticos también se merecen un agradecimiento por mi parte, ya que nunca habría llegado donde estoy (emocional y profesionalmente) sin ellos. Son unos cuantos, pero quiero hacer una mención especial a Marcos y a mi primi, Sara, por ser dos constantes en mi vida.

Por último quiero agradecer a mi familia más cercana el apoyo y ánimos recibidos para terminar la carrera, ya me los hayan proporcionado mediante apoyo y ánimos propiamente dichos, o mediante chantaje, meriendas, presión social y otras formas de extorsión igualmente efectivas. Un beso muy gordo desde estas líneas para papá, mamá, la tita, mis hermanos Manuel, Pilar, Belén y Eva y mis sobrinos Julia, Carmen, Irene, Marta, Adolfo, Pablo y Blanca. Gracias por tenerme en mente y preocuparos por mi.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Motivación del proyecto

El mundo de las interfaces Cerebro-Ordenador llama la atención quizás porque parece sacado de la ciencia ficción. La capacidad de controlar dispositivos empleando únicamente la mente puede tener muchas aplicaciones, desde devolver la capacidad comunicativa a personas que están totalmente aisladas debido a su incapacidad de mover cualquier músculo de su cuerpo, hasta el desarrollo de sistemas que *empaticen* con el usuario adaptando su funcionamiento al estado de ánimo del mismo, pasando por videojuegos que pongan a prueba el poder de control mental de los jugadores.

Es un privilegio poder trabajar en este ámbito tan interesante, y además disponer de un sistema BCI completo que no está al alcance de cualquiera, entre otras cosas porque se requieren ciertos instrumentos para captar las señales del cerebro que no están disponibles en cualquier entorno de trabajo.

Desde el grupo de investigación de Computación Evolutiva y Redes Neuronales de la Universidad Carlos III de Madrid (EVANNAI) [1], cuyas principales líneas de investigación son la Computación Evolutiva y el Aprendizaje Automático, se ha motivado el desarrollo de proyectos de fin de carrera sobre sistemas BCI. Javier Asensio, un estudiante de Ingeniería Informática de la Universidad Carlos III, leyó su proyecto de fin de carrera "Desarrollo de una Interfaz Cerebro-Máquina" [2] en Septiembre del año 2008. En su proyecto de fin de carrera diseñó, implementó y utilizó un sistema BCI completo que constaba de cierto hardware y software para la obtención de señales electroencefalográficas (desarrollado por la empresa *Brain Products GmbH* y del que disponía el grupo EVANNAI) y ciertas herramientas software diseñadas y desarrolladas por él mismo. Este sistema BCI es el punto de partida de este proyecto de fin de carrera.

La motivación de este proyecto de fin de carrera es doble. Por un lado se desea dar uso a este sistema BCI, y realizar experimentaciones con él, dado el interés que despierta trabajar con este tipo de sistemas. Se desean recopilar datos electroencefálicos y construir clasificadores con ellos que puedan utilizarse para controlar dispositivos mentalmente. Los datos recopilados además servirán para que se realicen investigaciones futuras, o podrán ser utilizados con fines docentes. Se podrían emplear en estudios sobre el tratamiento de señales electroencefalográficas para su uso en sistemas BCIs, ensayando diferentes técnicas para procesar la señal o para entrenar los clasificadores que se necesitan construir.

Por otro lado, la herramienta principal del sistema BCI (la aplicación MindReader) nació con el espíritu de ser ampliada y mejorada, y resulta de gran interés poder participar en su desarrollo y aportar algo nuevo a esta aplicación. Se desea añadir nuevas funcionalidades e integrar en ella funcionalidades del sistema BCI que se realizaban a través de otras aplicaciones.

1.2. Objetivos

En este proyecto de fin de carrera se tratarán de alcanzar dos objetivos principales:

1. **Adquisición y análisis de datos** mediante el sistema BCI existente.

   Se espera adquirir datos electroencefalográficos de diferentes personas mediante el sistema BCI disponible en el grupo de investigación EVANNAI de la universidad Carlos III de Madrid. Con estos datos se espera además entrenar clasificadores que, integrados en la aplicación MindReader, permitan que los voluntarios controlen el movimiento de un cursor por una pantalla. Podemos desglosar este objetivo en varios objetivos incrementales, donde la consecución de uno implica la posibilidad abordar el siguiente con éxito, y el fracaso en alcanzar alguno de ellos imposibilita continuar con el resto:

   a) Realización de sesiones de adquisición de datos con varios voluntarios para adquirir datos electroencefálicos.

   Se recopilarán gran cantidad de datos relativos a las señales encefálicas emitidas por varios voluntarios que se centrarán en realizar ciertos procesos mentales que se les indique.

   b) Análisis de los datos adquiridos, realizando diversos procesados a los mismos, para determinar si los datos contienen información útil para construir clasificadores.

Los datos obtenidos de cada voluntario serán procesados y analizados para poder determinar si un clasificador podría inferir características comunes de las señales emitidas al realizar un mismo proceso mental, de forma que pudiese clasificar señales electroencefálicas indicando qué proceso mental estaba realizando el usuario al emitir dicha señal.

c) Construcción de clasificadores que clasifiquen los datos adquiridos en las sesiones realizadas.

d) Realización de sesiones de adquisición de datos con retroalimentación empleando los clasificadores obtenidos anteriormente. En estas sesiones se obtendrán más datos electroencefálicos, además de comprobar la capacidad de clasificación en tiempo real de los clasificadores.

e) Mejora de los clasificadores obtenidos hasta el momento, empleando los datos obtenidos en las sesiones de adquisición con retroalimentación para reentrenar los clasificadores.

f) Realización de sesiones de clasificación en las cuales los voluntarios controlan la actividad de un cursor a través de la modulación de sus procesos mentales.

2. **Ampliación de la herramienta MindReader** desarrollada en el contexto del proyecto de fin de carrera de Javier Asensio [2].

   La aplicación MindReader es la aplicación principal de sistema BCI utilizado en este proyecto de fin de carrera. El segundo objetivo principal de este proyecto es mejorar dicha herramienta, añadiendo nuevas funcionalidades e integrando las funcionalidades ya existentes en otros programas auxiliares que emplea el sistema BCI (como la aplicación nnt) en la aplicación principal del sistema.

   Este objetivo puede desglosarse en los siguientes objetivos, independientes entre sí:

   a) Integrar las funcionalidades existentes en el programa nnt en el programa MindReader.

   b) Permitir que se apliquen filtros a los datos adquiridos y que éstos puedan seleccionarse de forma dinámica. (Nueva funcionalidad).

   c) Permitir el simulado de sesiones de adquisición a partir de datos almacenados en un fichero, de forma que pueda utilizarse la herramienta MindReader sin necesidad de conectarla al servidor de

datos *Brain Vision Recorder*. Además de esta forma podrán aplicarse diferentes procesados de datos a los datos de una determinada sesión de adquisición, posteriormente a su realización. (Nueva funcionalidad).

d) Permitir el reentrenamiento de redes de neuronas artificiales de tipo perceptrón multicapa (MLP), las que se generan con el sistema existente. Es decir, que se pueda tomar un clasificador de tipo red de neuronas MLP, y volver a realizar un entrenamiento del mismo con nuevos patrones, para que se reajusten los pesos de las conexiones neuronales. Esto permitirá que se realicen entrenamientos incrementales de clasificadores. (Nueva funcionalidad).

e) Detectar otras posibles carencias de la herramienta y tratar de solventarlas.

1.3. Contenido de la memoria

En este capítulo se detalla el contenido de la memoria de este proyecto de fin de carrera.

Capítulo 1. Introducción En este capítulo se hace una breve introducción al proyecto (apartado 1.1) y se comentan los objetivos planteados al inicio del mismo (apartado 1.2). Además se indica el contenido de este documento (apartado 1.3).

Capítulo 2. Interfaz Cerebro-Ordendador (BCI) En este capítulo se hace una introducción a los sistemas BCIs. En el apartado 2.1 se comentan las posibles aplicaciones prácticas que tienen los sistemas BCIs. En la sección 2.2 se repasan brevemente los aspectos básicos de la anatomía del encéfalo (apartado 2.2.1), se mencionarán los procesos mentales asociados con cada área del cerebro (apartado 2.2.2) y la clasificación de las señales electroencefalográficas (apartado 2.2.3). En la sección 2.3 se explica cómo los sistemas BCIs utilizan el aprendizaje automático y el *feedback*, también se comenta en qué consiste el fenómeno denominado "BCI-Illiteracy". En la sección 2.4 se hace una revisión extensiva de las diferentes técnicas que usan los sistemas BCIs para captar las señales encefálicas, a la vez que se enumeran experimentos e investigaciones que se están llevando a cabo en la actualidad o que han sido realizadas recientemente. Se ha separado esta sección según los sistemas empleen técnicas no invasivas (apartado 2.4.1) o técnicas invasivas (apartado 2.4.2). Por último, en la sección 2.5 se comentan las transformaciones

más comunes que se aplican a los datos electroencefalográficos antes de utilizarlos para construir un clasificador.

Capítulo 3. Hardware y software implicado En este capítulo se explican qué elementos hardware y software fueron utilizados en las sesiones de adquisición de datos realizadas y en el análisis posterior de los datos obtenidos.

Capítulo 4 Ampliación de la Herramienta En este capítulo se comentan todo lo relativo a las modificaciones realizadas a la aplicación MindReader. En el apartado 4.1 "Punto de Partida:primera versión del programa MindReader y nnt" se comentan las características de la primera versión del sistema BCI (funcionalidades, ficheros de salida, procesadores de datos utilizados y clasificadores, parámetros utilizados y algunos detalles de la implementación). En el apartado 4.2 "Aspectos a mejorar de la herramienta" se enumeran algunas características del programa que deberían ser modificadas. En el apartado 4.3 se comentan las mejoras realizadas a la herramienta. Por último, en el apartado 4.4 "Nuevas funcionalidades. Entrenamiento redes-Simulación sesiones." se explica detalladamente las nuevas funcionalidades añadidas a las que se accede a través de la nueva interfaz de ventana de *Entrenamiento de Redes y simulación de sesiones*, indicando cómo se usan, los ficheros de salida que se generan, ciertas características relativas a la usabilidad y algunos detalles sobre la implementación realizada.

Capítulo 5 Adquisición de datos En este capítulo se comentan los aspectos relativos a la adquisición de datos realizada durante el desarrollo de este proyecto de fin de carrera. Se comentan los detalles de cada sesión de adquisición realizada y su preparación (apartado 5.1), los parámetros empleados en las mismas (apartado 5.2), y las decisiones tomadas a la hora de procesar y analizar los datos y los resultados obtenidos en los análisis realizados (apartado 5.3.

Capítulo 6 Conclusiones y Trabajos Futuros En este capítulo se analizan los objetivos alcanzados y se recogen las posibles líneas futuras de ampliación y mejora del sistema.

Apéndice A Guía de Instalación Manual de instalación y desinstalación de la aplicación.

Apéndice B Presupuesto Presupuesto del desarrollo del proyecto.

Apéndice C Contenido de los DVDs Contenido de los dos DVDs que contienen los resultados de este proyecto de fin de carrera.

Anexo: D Acrónimos y Abreviaturas Listado de acrónimos y abreviaturas utilizados a lo largo del documento.

Capítulo 2

Interfaz Cerebro Ordenador (BCI)

En la última década ha aumentado considerablemente el interés en el desarrollo de interfaces de comunicación eficaces que permitan conectar el cerebro humano con un ordenador. El desarrollo de este tipo de sistemas requiere de la investigación de equipos multidisciplinares de científicos e ingenieros, implicando expertos en fisiología, ciencia de los materiales, instrumentación y procesamiento de señales, inteligencia artificial, mecánica, electrónica, robótica e informática, entre otras.

La conexión puede ser en un sentido (comunicación desde el cerebro humano a la máquina, llamada BCI, acrónimo del término inglés *Brain-Computer Interface*, Interfaz Cerebro-Ordenador) o en el otro (a veces denominados *Computer-Brain Interface* para diferenciarlos de los anteriores, donde la comunicación parte de la máquina al cerebro del sujeto).

Un ejemplo de este último caso es el implante coclear, un aparato electrónico que puede implantarse en pacientes con sordera neurosensorial severa a profunda que no responden favorablemente a los audífonos convencionales. Consta de una parte externa con un micrófono y un procesador del lenguaje, y otra parte que se implanta en el oído interno conectando varios electrodos al nervio coclear. El micrófono recoge el lenguaje y demás sonidos, y el procesador del lenguaje selecciona la parte más importante de la información sonora y luego produce un patrón de pulsos eléctricos en el oído interno del paciente. Los electrodos emulan la función de miles de células ciliadas de un oído normal, el mensaje se envía así al cerebro y se interpreta como un sonido. [3] También se está investigando y ya se han realizado algunas pruebas de implantes de retina artificial, un sistema que suple las funciones de procesamiento de imágenes de la retina y transmite los resultados al cerebro a través de un conjunto de electrodos.[4] Otra línea de investigación se

centra en sistemas que permiten estimular mediante señales eléctricas artificiales ciertas zonas del cerebro de un paciente con Parkinson y que podrían disminuir la sintomatología del sujeto. [5]

En esta sección nos centraremos en dar una visión general del otro tipo de sistemas, ya que en este proyecto de fin de carrera se utiliza y se mejora un sistema BCI donde se pretende que sea el usuario con sus pensamientos el que active las funciones de una máquina.

En primer lugar, en la sección 2.1 se comentan las posibles aplicaciones que tienen los sistemas BCIs. En la sección 2.2 se repasarán brevemente los aspectos básicos de la anatomía del encéfalo, la clasificación de las señales electroencefalográficas, y se mencionarán los procesos mentales asociados con cada área del cerebro. El repaso de estos conceptos es conveniente ya que son muy utilizados en el ámbito de los sistemas BCIs. En la sección 2.3 se explica cómo los sistemas BCIs utilizan el aprendizaje automático y el *feedback* para conseguir alcanzar el objetivo de la comunicación eficaz humano-máquina mediante el uso de señales cerebrales, en esta sección también se comenta en qué consiste el fenómeno denominado "BCI-*Illiteracy*". En la sección 2.4 se hace una revisión extensiva de las diferentes técnicas que usan los sistemas BCIs para captar las señales encefálicas, a la vez que se enumeran experimentos e investigaciones que se están llevando a cabo en la actualidad o que han sido realizadas recientemente. Por último, en la sección 2.5 se comentan las transformaciones más comunes que se aplican a los datos electroencefalográficos para que puedan utilizarse para construir clasificadores que determinen en qué está pensando un sujeto en cada momento.

2.1. Aplicaciones BCI

Una de las posibles aplicaciones de los sistemas BCIs es mejorar la calidad de vida de pacientes con deficiencias físicas neuromusculares graves que conservan su habilidades cognitivas intactas. Los pacientes con el síndrome *lock-in* completo ("enclaustramiento") sufren de parálisis de todos sus músculos voluntarios, incluido el movimiento ocular. Para este tipo de pacientes es de vital importancia el desarrollo de sistemas BCI que les permitan comunicarse con el mundo exterior.

Una interfaz cerebro-ordenador permitiría manejar ordenadores, sillas de ruedas, prótesis y otros dispositivos simplemente con señales emitidas por el cerebro.

También puede usarse como herramienta de trabajo en el ámbito de la neurociencia, facilitando la comprensión del funcionamiento del cerebro humano: cómo coordina el comportamiento humano, cómo se adquiere un nuevo

comportamiento y cómo se mantiene. Un sistema BCI permite investigar la actividad cerebral como una variable independiente, algo novedoso en este contexto. En experimentos tradicionales, se indica al usuario que realice una tarea o se le somete a algún estímulo (variables independientes) y se mide la actividad cerebral como variable dependiente de la anterior. Con un sistema BCI el sujeto de experimentación puede aprender ayudado por la retroalimentación a incrementar o decrementar deliberadamente la actividad cerebral (variable independiente) y las variaciones en su comportamiento pueden medirse como variable dependiente de lo anterior.

Por último no podemos olvidar mencionar su uso como nuevo paradigma de interacción entre las personas y las máquinas, que ofrecería nuevas opciones de interacción y control del dispositivo. En personas sanas no tiene mucho sentido utilizar un sistema BCI para comunicarse, ya que existen muchos otros métodos más eficaces y menos costosos. No obstante sí que sería muy interesante que el dispositivo pueda detectar ciertos estados mentales del sujeto (como el nivel de alerta o su estado emotivo) y de esta forma adecuar su funcionamiento a dicho estado del usuario.

2.2. Encéfalo: anatomía, funciones cerebrales y señales encefálicas.

2.2.1. Anatomía del Encéfalo

El sistema nervioso central es una estructura extraordinariamente compleja que recoge millones de estímulos por segundo que procesa y memoriza continuamente, adaptando las respuestas del cuerpo a las condiciones internas o externas. El encéfalo es la estructura más importante del sistema nervioso central. Está protegido por los huesos del cráneo, en la cavidad craneana, y por tres membranas denominadas meninges. Las meninges envuelven por completo el neuroeje (encéfalo y médula espinal), interponiéndose entre éste y las paredes óseas. De afuera hacia adentro, las meninges se denominan duramadre, aracnoides y piamadre.[6]

El encéfalo consta de tres partes: el cerebro, el cerebelo y el tronco cerebral. Más específicamente, el encéfalo consta de los siguientes elementos que se enumeran a continuación, mostrados además en la figura 2.1 [7][8] [9]:

- *Protuberancia* Parte profunda del encéfalo que se encuentra en el tronco cerebral y que contiene muchas de las áreas encargadas del control de los movimientos oculares y faciales.

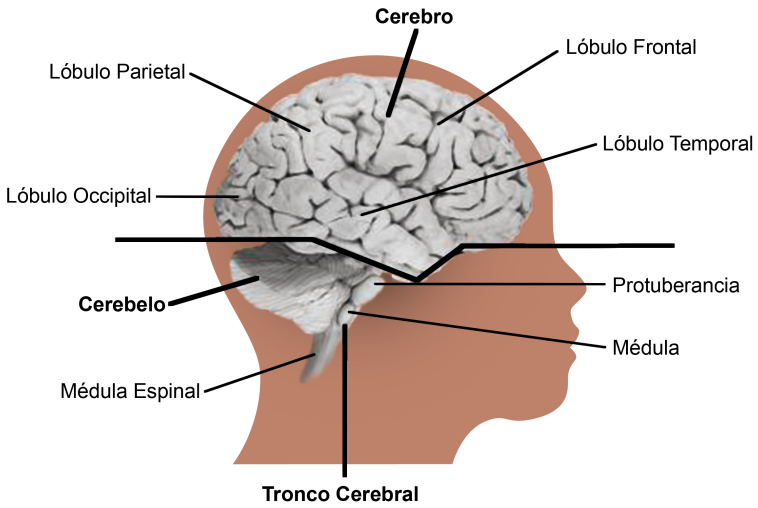

Figura 2.1: División del Encéfalo

- *Médula* Parte más baja del tronco cerebral; es la porción más importante del encéfalo y contiene los centros fundamentales del control de las funciones cardíacas y pulmonares.

- *Médula espinal* No forma parte del encéfalo propiamente dicho, pero se extiende desde la base del encéfalo hasta la región lumbar. Es un haz largo de fibras nerviosas que se encuentra en la espalda, encargado de conducir los mensajes entre el encéfalo y el resto del cuerpo.

- *Lóbulo frontal* Porción más voluminosa del encéfalo, se encuentra en la región anterior de la cabeza; interviene en las características de la personalidad y en el movimiento. Está asociado con las funciones mentales superiores: pensar, planificar, decidir.

- *Lóbulo parietal* Parte media del encéfalo que ayuda a una persona a identificar objetos y a comprender las relaciones espaciales (dónde se encuentra nuestro cuerpo con relación a los objetos que nos rodean). El lóbulo parietal también interviene en la interpretación del dolor y del tacto en el cuerpo.

- *Lóbulo occipital* Parte posterior del encéfalo que interviene en la visión.

- *Lóbulo temporal* Los lados del encéfalo o lóbulos temporales intervienen en la memoria, el habla y el sentido del olfato.

2.2.2. Funciones de la corteza cerebral.

No es posible hablar con rigor de centros cerebrales donde se asientan las funciones mentales, porque actúan solapadamente entre sí. Cualquier aproximación descriptiva de las funciones cerebrales y su localización en la corteza pueden resultar reduccionista. No obstante, es necesaria para entender la actividad que lleva a cabo la corteza cerebral. A continuación se enumeran en qué procesos cognitivos participa cada lóbulo. En la figura 2.2 se muestra la correspondencia entre las funciones cerebrales y los lóbulos cerebrales.[11][12]

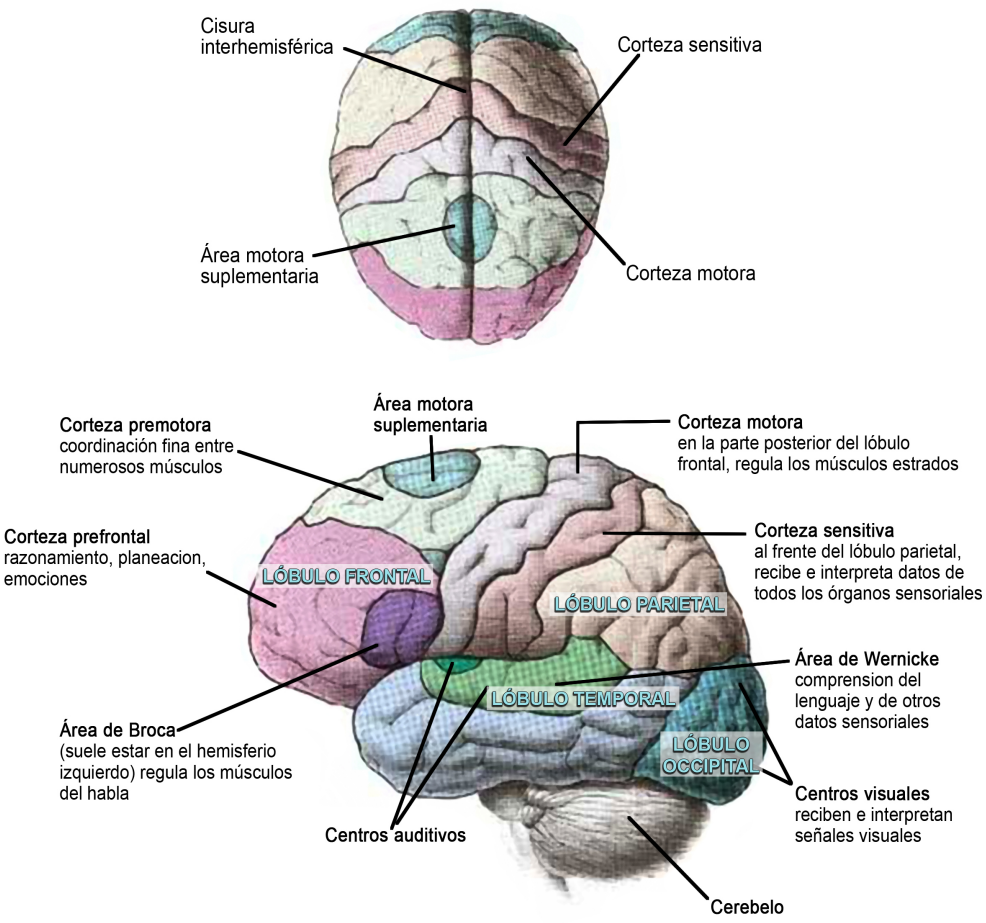

Figura 2.2: Lóbulos encefálicos y funcionalidades

- *Lóbulo Occipital* Principalmente se encarga del procesamiento de la información para el color, la forma y el movimiento. Va a intervenir también en todos aquellos procesos cognitivos que requieren de la participación de la visión :

  - Procesamiento de información visual útil para dirigir movimientos específicos.
  - Contribuye al rastreo visual, en la búsqueda de las características importantes y distintas.
  - Reconocimiento visual (objetos, rostros, etc.)
  - Ayuda al procesamiento de relaciones espaciales
  - Atención visual

- *Lóbulo Parietal* La zona anterior procesa sensaciones y percepciones somáticas, la posterior está especializada en integrar información proveniente de las vías sensitivas para el control del movimiento. El lóbulo parietal esta implicado en extracción y manejo de información espacial. De esta manera interviene en aquellos procesos en los que el manejo de información espacial es necesaria, es decir:

  - Reconocimiento de objetos (localización y movimiento)
  - Guía del movimiento

  También interviene en otras actividades complejas como son: la rotación y manipulación mental de objetos, conceptualización de la derecha y la izquierda o en la resolución de problemas aritméticos dependientes de representación espacial de los números (Ej. Multiplicaciones).

- *Lóbulo Temporal* Subcorticalmente contiene las cortezas auditiva primaria, auditiva y visual secundarias, la corteza límbica, la amígdala y el hipocampo. Interviene en:

  - Procesamiento de los estímulos auditivos (incluyendo el lenguaje)
  - Reconocimiento de objetos visuales (forma, color, tamaño)
  - Almacenamiento a largo plazo (memoria).
  - Emoción

- *Lóbulo Frontal* Funcionalmente podemos distinguir tres áreas: corteza motora, premotora y prefrontal.

La principal función de la corteza motora es la de proporcionar los movimientos (incluyendo las secuencias motóricas necesarias para el habla o los movimientos oculares). La corteza premotora es la encargada de seleccionar los movimientos que van a ser ejecutados y finalmente es la corteza prefrontal la responsable de la ejecución de los movimientos.

Ademas la corteza prefrontal interviene en:

- Procesos atencionales y mnésicos (recuerdos).
- Formación de conceptos, operaciones formales, flexibilidad cognitiva.
- Formulación de metas, planificación de pasos, monitorización de la ejecución.
- Toma de decisiones, resolución tareas novedosas, conducta social, juicio ético.

2.2.3. Señales electroencefálicas

La propagación de las señales eléctricas por las neuronas provoca corrientes eléctricas (iónicas) en el medio extracelular y potenciales extracelulares que pueden ser detectados mediante un electroencefalógrafo (EEG) o un magnetoencefalógrafo (MEG). El potencial eléctrico detectado se debe a la combinación de los potenciales eléctricos de numerosas sinapsis dendríticas de neuronas que presentan una orientación regular.[11] (Ver figura 2.3 [10])

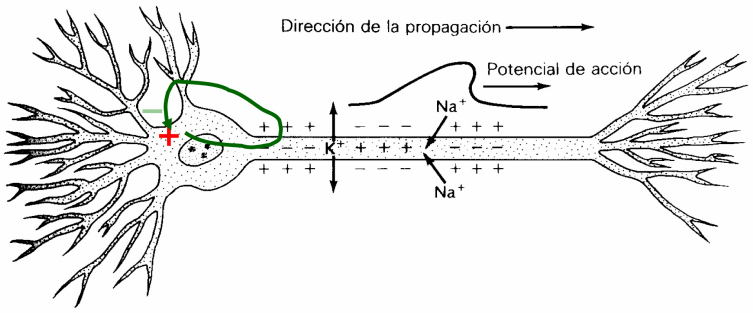

Figura 2.3: Propagación señales eléctricas en la neurona

Normalmente a las señales obtenidas mediante EEG se distinguen en cuatro clases según la frecuencia dominante: *ondas Beta* (frecuencia mayor a 13 Hz), *ondas Alfa* (frecuencia entre 8Hz y los 13Hz), *Ondas Theta* (frecuencia entre los 4 Hz y los 7 Hz) y *ondas Delta* (frecuencia inferior a 4Hz). En

la figura 2.4 se muestra esta clasificación y en la figura 2.5 se muestran el aspecto que toman dichas ondas en el EEG o en el MEG.[10])

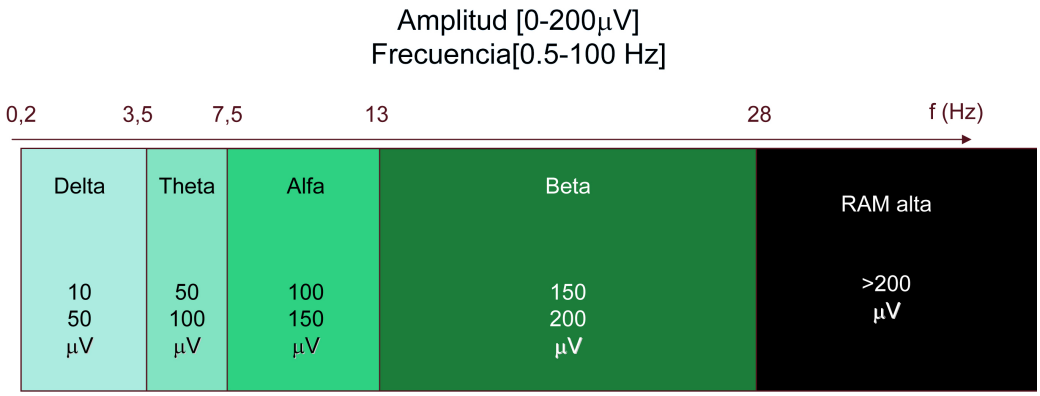

Figura 2.4: Clasificación de ondas cerebrales.

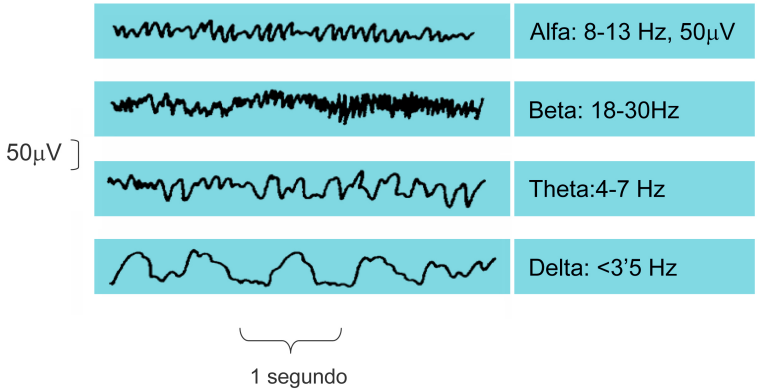

Figura 2.5: Ejemplos ondas cerebrales

Cada una de ellas tiene una correspondencia con el estado en el que se encuentra el sujeto que las emite. En la figura 2.6 se muestra dicha correspondencia. En la figura 2.7 se muestra cómo al abrir y cerrar los ojos se pasa del ritmo beta al alfa y viceversa. [10])

Aparte de estás bandas específicas de frecuencia, son conocidas otras ondas cerebrales típicas por su forma (espiga, onda lenta, onda aguda...). También es importante reseñar la existencia de potenciales evocados, estas señales son patrones reconocibles derivados de un estímulo somatosensorial.

Unas ondas muy interesantes para el ámbito de los sistemas BCIs son los Potenciales Cerebrales Relacionados con el Movimiento (PCRMs), en

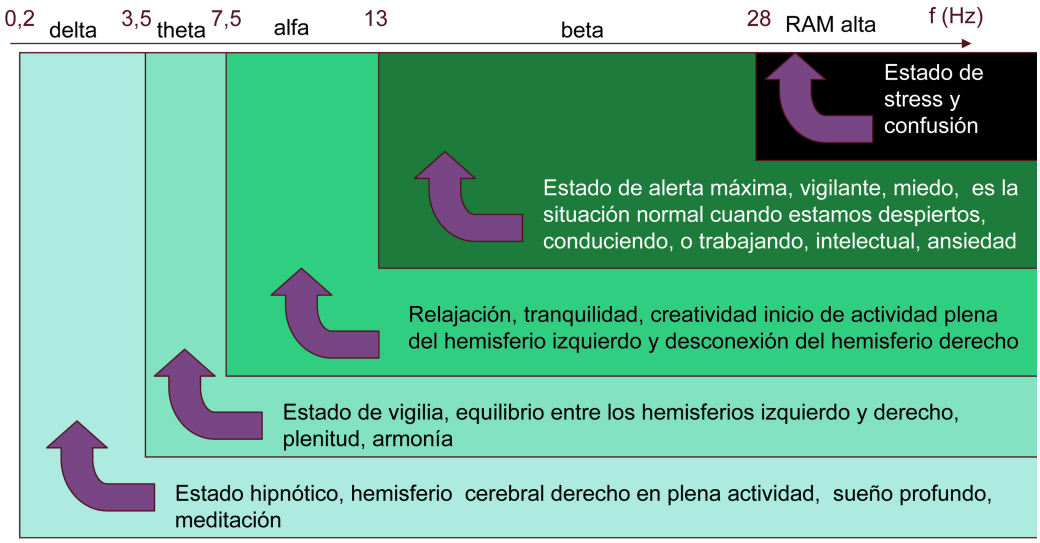

Figura 2.6: Correspondencia entre ondas cerebrales y estado mental

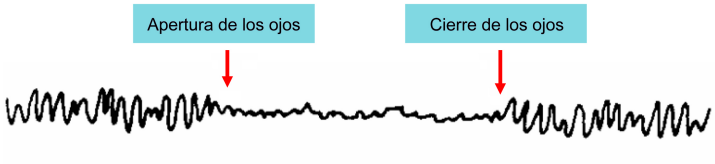

Figura 2.7: Sustitución del ritmo alfa por el beta al abrir los ojos

inglés *Movement-Related Cortical Potentials (MRCP)*. El estudio de estos potenciales se centra en las señales registradas antes de que se produzca un movimiento motor voluntario, sin necesidad de que el movimiento llegue a producirse. Pueden aparecer incluso hasta un segundo antes de que se produzca el movimiento.[12] La existencia de estos potenciales nos permite plantearnos la posibilidad de detectar automáticamente la intención de realizar un movimiento aunque éste no se lleve a término. Además estos potenciales se producen en zonas localizadas de la corteza motora, es decir, podríamos identificar la señal y en qué zona ocurre.

En la sección 2.4 se mencionarán y dará una breve explicación de alguna que otra señal característica más que puede emplearse en sistemas BCIs, como por ejemplo la onda P300.

2.3. Aprendizaje Automático y Retroalimentación (*feedback*). BCI-Illiteracy.

Encontramos dos opciones para conseguir que una máquina y una persona se comuniquen entre sí a través del cerebro de la persona. En realidad estos dos enfoques se combinan en casi todos los sistemas BCIs existentes.

La primera opción es que la persona aprenda a modular conscientemente y de forma voluntaria las señales que emite para que la máquina pueda "comprenderlas". Cuando el usuario ha aprendido a alcanzar de forma voluntaria ciertos estados de actividad cerebral que la máquina es capaz de detectar, los diferentes estados pueden usarse para invocar las funciones de la máquina. Para lograr esto el sujeto recibe biorretroalimentación (*biofeedback* en inglés), información en tiempo real de cierto parámetro (normalmente relativo a su actividad cerebral, aunque también puede ser sobre su actividad cardiovascular, temperatura, conductividad de la piel...) y debe intentar incrementar o decrementar la actividad en cuestión según se le vaya indicando. Se informa al usuario de los resultados obtenidos después de cada intento, y pueden utilizarse técnicas de condicionamiento para facilitar el aprendizaje, como el refuerzo positivo (cuando el sujeto logra realizar la tarea se muestra una cara sonriente, o suma puntos que podrían canjearse por obsequios o dinero, etc). Tras varias sesiones de entrenamiento los participantes logran modular cierta actividad cerebral de forma voluntaria.

La segunda opción es que la máquina pueda detectar automáticamente la señal del sujeto sin necesidad de entrenamiento por parte del mismo, empleando algún algoritmo de aprendizaje automático avanzado (redes de neuronas artificiales, máquinas de soporte vectorial...) El algoritmo de aprendizaje debe adaptarse al usuario que va a realizar la tarea, se necesitan ejemplos para que el algoritmo pueda inferir las características estadísticas de las señales que emite el sujeto. Por lo tanto, aunque no se requiera de entrenamiento por parte del sujeto para aprender a modular su actividad cerebral, sí que se necesita que realice una o varias sesiones de calibración para "entrenar" al algoritmo que utilizará la máquina. En estas sesiones se pide al sujeto que realice ciertos procesos mentales repetidas veces, y con los datos obtenidos se construye el algoritmo. El mayor problema que encontramos en este enfoque es la alta variabilidad de los datos entre sesiones, por lo que es imprescindible el uso de algoritmos de aprendizaje automático avanzado. Una gran ventaja de adoptar este enfoque es que el uso de estos algoritmos como clasificadores no sólo permite construir el sistema BCI, el clasificador detecta características que no son apreciables con otras técnicas más simples y el análisis de los datos a través de este tipo de algoritmos permite ampliar

conocimientos del ámbito de la neurociencia.

Hay que tener en cuenta que para que los algoritmos de aprendizaje automático obtengan resultados satisfactorios, generalmente debe realizarse cierto procesado de los datos antes de emplearlos como patrones (ejemplos) para construir el clasificador. Por ejemplo, la aplicación MindReader, empleada en este proyecto de fin de carrera, recibe los datos de la señal electroencefálica en el dominio del tiempo. La señal se pasa al dominio de la frecuencia, calculando una estimación de la densidad espectral de potencia (PSD) empleando el método de Welch. Las bases teóricas del PSD y el método de Welch para el cálculo del PSD ya fueron ámpliamente explicadas en la memoria del proyecto de fin de carrera de Javier Asensio [2] y se mencionarán brevemente en el apartado 2.5.

Como se ha comentado anteriormente, estos dos enfoques opuestos suelen emplearse de forma conjunta. Por ejemplo, en el Berlin Brain-Computer Interface (BBCI) [13], cuya estructura es muy similar a la del sistema BCI empleado en este proyecto, no se emplea *biofeedback* propiamente dicho, ya que en ningún momento se le muestra al usuario el electroencefalograma que se está generando en tiempo real. Pero sí que se proporciona al usuario cierta retroalimentación (*feedback*), al menos en la segunda fase de calibración de la herramienta. En una primera fase se le indica al usuario que realice ciertos procesos mentales y con estos datos se genera un clasificador. En la segunda fase, además de indicar al usuario el proceso mental a realizar, se indica el resultado que hubiese obtenido el clasificador anteriormente entrenado al intentar clasificar los datos que se están obteniendo en tiempo real. De esta forma el usuario sabe cuando la máquina clasificaría correctamente sus pensamientos y cuándo no, y adaptará la forma de realizar dichos procesos mentales repitiendo las pautas que obtienen buenos resultados. Luego el usuario también se está adaptando a la máquina, no sólo la máquina se adapta al usuario.

Aunque no todos los sistemas BCI utilicen *biofeedback*, prácticamente todos ellos proporcionan retroalimentación al usuario sobre el éxito/fracaso de la tarea que está realizando. El *feedback* puede proporcionarse de forma visual, auditiva, táctil... Un caso curioso es el de un sistema BCI desarrollado por Birmaumer y Cohen en el año 2005, se colocaba un guante protésico a pacientes con parálisis en la mano. Realizar el proceso mental de imaginar el movimiento de la mano abría el guante (y la mano del paciente) y relajarse lo cerraba, obteniendo los pacientes el *feedback* directamente de su miembro paralizado.[5]

Por último, comentar que los sistemas BCI no son capaces de clasificar satisfactoriamente los patrones de activación cerebral de aproximadamente un 20 % de los usuarios, independientemente de que se utilicen técnicas de

aprendizaje automático, *feedback* o una combinación de ambas. Se necesitará realizar más investigaciones al respecto para poder comprender en su totalidad a qué es debido este fenómeno, denominado en inglés *BCI-Illiteracy*.

2.4. Sistemas Invasivos vs No Invasivos. Técnicas Utilizadas

Existen gran variedad de sistemas BCIs, que difieren en la actividad neuronal que recogen, cómo se entrenan los sujetos o como las señales son traducidas a comandos para controlar dispositivos. En esta sección se quiere dar una visión general de los distintos sistemas BCIs existentes, partiendo del sistema de obtención de la señal.

Se pueden clasificar los sistemas BCIs en invasivos (que requieren el implante de electrodos en el cerebro) o no invasivos.

Los primeros sistemas BCI que se desarrollaron estaban basados en la adquisición de datos electroencefalográficos mediante técnicas no invasivas. La principal ventaja de estos sistemas es que al no requerir implantes quirúrgicos, se amplía significativamente el rango de posibles aplicaciones del sistema. Su principal desventaja es que la señal que puede adquirirse con este tipo de dispositivos es de intensidad muy baja, sin embargo los avances en las técnicas de procesado de señal y el desarrollo de electrodos secos hacen este enfoque muy atractivo.

Las técnicas invasivas suelen tener más éxito que las no invasivas.

2.4.1. BCIs no invasivos

Señales cerebrales adquiridas mediante electroencefalograma (EEG-BCIs)

La actividad eléctrica del cerebro puede ser recogida mediante electroencefalografías. Sus principales ventajas son que no requiere cirugía al ser un método no invasivo, que es económicamente asequible, y se puede usar en el entorno del paciente al ser fácilmente transportable.

Las señales registradas por la EEG se ven afectadas por los diferentes grados de resistencia de los tejidos que traspasan hasta alcanzar el electrodo externo, lo que conlleva dificultades e imprecisiones al interpretar la localización de las diferentes fuentes cerebrales generadoras de la señal electroencefalográfica.

Además, en el caso de los electrodos pasivos debe usarse gel conductivo (aunque existen electrodos pasivos secos y electrodos activos que no lo

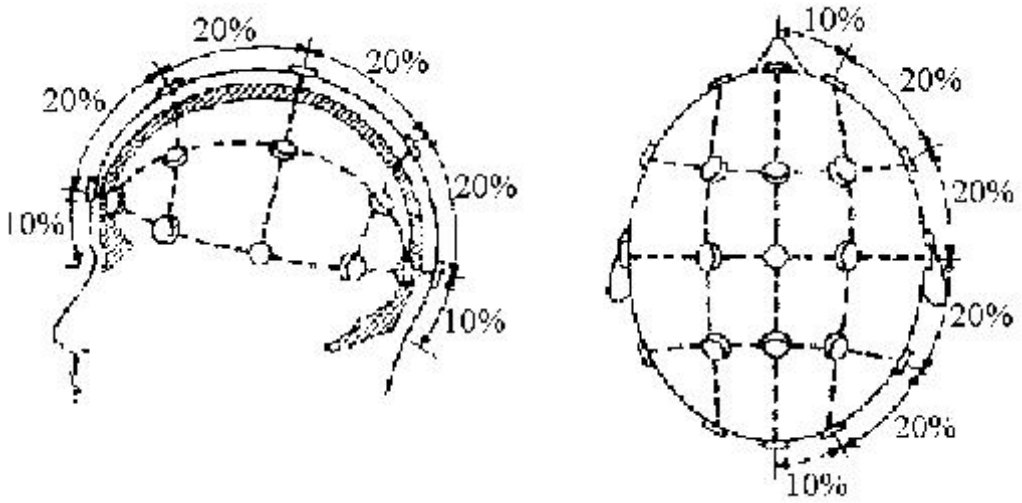

Figura 2.8: Sistema Internacional 10-20 de colocación de electrodos)

requieren) y la colocación del casco EEG requiere demasiado tiempo.

Uno de los métodos más extendido para ubicar y nombrar los electrodos en un casco encefalógrafo es el sistema internacional 10-20. Los electrodos se colocan sobre la esfera imaginaria que mejor se ajuste al cráneo del sujeto. El 10 ó 20 indica que los electrodos están colocados a un 10 % o un 20 % de la distancia entre el el inion y el nasion (ver figura 2.8). Los electrodos se nombran mediante unas iniciales, que indican el lóbulo en el que se ubica el electrodo, y un número, que indica su posición en el hemisferio. Las letras F, T, C, P y O son abreviaturas de Frontal, Temporal, Central, Parietal y Occipital respectivamente (nótese que aunque el lóbulo central no exista, se utiliza la letra "C" con fines identificativos). Una "z" (de *zero*) se refiere a electrodos colocados en la línea central. Los números pares (2,4,6,8) se refieren a electodos colocados en el hemisferio derecho, mientras que los impares (1,3,5,7) indican que el electrodo está situado en el hemisferio izquierdo. La numeración aumenta según los electrodos están más alejados de la línea central. Si se utilizan más electrodos de los que permite representar este sistema, se ubican entre el espacio de estos electrodos, nombrándose con las iniciales de los electrodos entre los que se encuentra y el número que corresponda con su ubicación, esto se conoce como el sistema 10-20 extendido (ver figura 2.9).[14]

Los rasgos específicos del EEG pueden ser regulados por el usuario (como los potenciales corticales lentos o los ritmos sensoriomotores) o bien provocados por un estímulo visual, auditivo o táctil (como es el caso de los potenciales

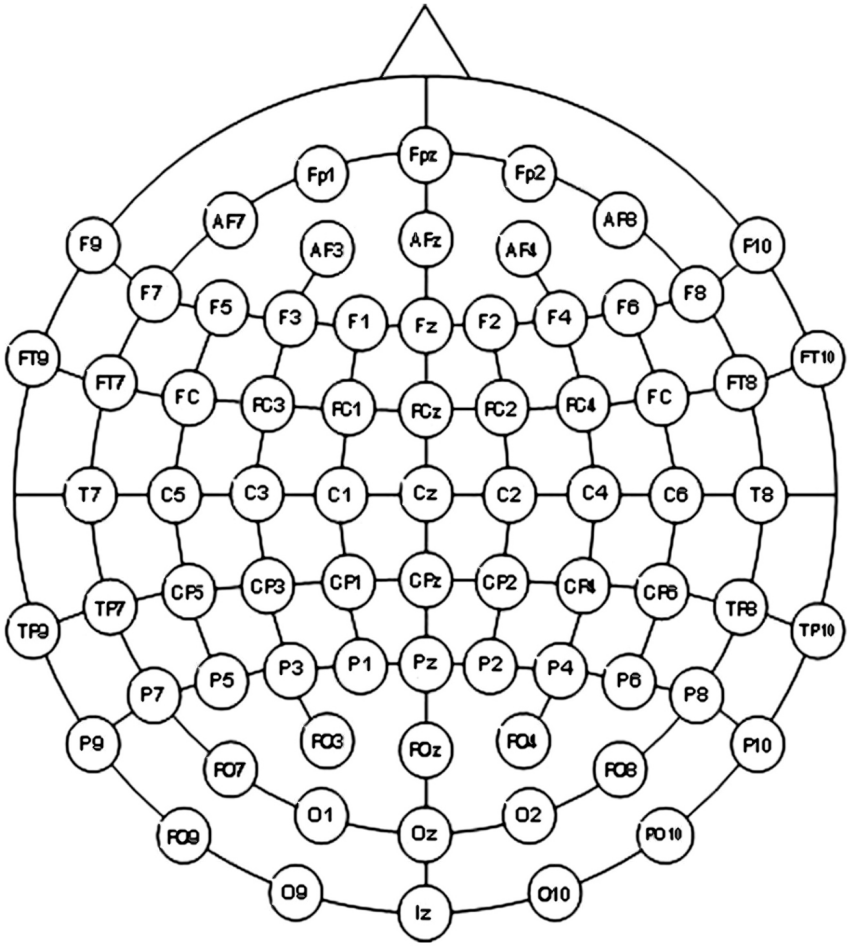

Figura 2.9: Sistema Internacional 10-20 de colocación de electrodos extendido

evocados). A continuación se comenta la fisiología de estos rasgos y su uso en sistemas EEG-BCIs.

- *Potenciales corticales lentos (PCL)*

  Los potenciales corticales lentos, en inglés *slow cortical potentials* (SCP), son cambios lentos (el periodo de variación se encuentra entre 0.5 y 10 segundos) en la amplitud de frecuencia en el electroencefalograma. Un sujeto entrenado puede aprender a regularlos y controlar así un sistema BCI. Estos sistemas requieren bastante tiempo de entrenamiento por parte del sujeto (semanas a meses). En las primeras

sesiones de entrenamiento el sujeto recibe retroalimentación continua acerca de los cambios corticales reales, por ejemplo de forma visual mediante un punto que cambia de tonalidad según la amplitud de la señal cortical. Posteriormente se pueden realizar sesiones de entrenamiento sin retroalimentación, en las cuales se mantiene e incluso se mejora la habilidad de controlar los PCLs.[15]

- *Potenciales relacionados con eventos, onda P300*

  Los potenciales relacionados con eventos o potenciales evocados, en inglés *event-related potentials* (ERPs), son potenciales electrocorticales que pueden ser medidos mediante un EEG antes, durante o después de un evento sensorial (e.g. recepción de un estímulo visual), psicológico (e.g. realizar determinado pensamiento) o motor (e.g. movimiento de una mano). La onda P300 es un potencial relacionado con eventos que puede ser registrado mediante EEG como una deflexión positiva de voltaje con una latencia de unos 300ms en el EEG desde la aparición del estímulo visual o auditivo [16] [5]. Esta señal aparece normalmente cuando se somete a un sujeto a un estímulo excepcional dentro de una serie de estímulos típicos.

  En los años ochenta Farwell y Donchin construyeron un dispositivo que permitía seleccionar un elemento de entre los 36 contenidos en una matriz 6x6. Los participantes fijaban su atención en uno de los elementos. Cada 100 ms se iluminaba una fila o una columna de la matriz. Cuando se han iluminado las seis filas y las seis columnas de la matriz, el elemento en el que el sujeto se fija sólo ha parpadeado dos veces (cuando se iluminó la fila y la columna donde está situado). Este simple sistema BCI determina su salida detectando la fila y la columna que al iluminarse provocaron en el sujeto la onda P300 de mayor amplitud. Una gran ventaja de este tipo de sistemas es que su uso no requiere de mucho entrenamiento por parte del sujeto ya que no es necesaria la autorregulación del EEG, aunque en una primera fase sí que haya que calibrar el sistema para detectar cual es la actividad normal del cerebro para poder detectar posteriormente la onda P300. No obstante, un inconveniente importante es que requiere que el sujeto tenga el sentido de la vista intacto y pueda fijar su atención y mirar un elemento determinado, quizás por este motivo se están realizando investigaciones sobre cómo trasladar este modelo empleando estímulos auditivos en lugar de visuales [5].

- *Potencial evocado visual continuo*

Tras la estimulación visual se produce un potencial evocado que puede ser registrado desde el córtex visual, en el lóbulo occipital. Los estímulos intermitentes de frecuencia variable (entre 2 Hz y 15 Hz) dan origen a un potencial evocado visual continuo, en inglés *Steady-state Visual Evoked Potential* (SSVEP), que se caracteriza por un ritmo EEG cuya frecuencia es la misma que la del estímulo. Esto permite detectar la presencia de un SSVEP analizando las componentes de frecuencia del EEG registradas sobre esta zona. Los BCIs que se aprovechan de esta característica presentan distintos parpadeos (previamente asociados a acciones) simultáneamente. El sujeto puede ejecutar un comando deseado mirando el estímulo asociado.

Como ejemplo se puede mencionar el BCI desarrollado por Middendorf y colaboradores, que permite seleccionar botones de la pantalla. En promedio, los 8 sujetos involucrados en el experimento alcanzaron una rapidez de 2.1 segundos por selección, y un porcentaje de 92 % de precisión sin requerir prácticamente de entrenamiento previo. [17]

Al igual que los sistemas que detectan la onda P300, los sistemas BCIs que emplean SSVEP requieren que el sentido de la vista del paciente esté intacto, pero tiene la ventaja que no requiere que se calibre el sistema.

- *Ritmo Sensoriomotor*

  El ritmo sensoriomotor, en inglés *sensorimotor rythm* (SMR), decrementa o se desincroniza al realizar un movimiento voluntario e incrementa o se sincroniza en la fase posterior al movimiento o durante la relajación. Lo más importante para su aplicación en el control de sistemas BCIs es que la desincronización del ritmo sensoriomotor también se produce al imaginarse el movimiento, no sólo al realizarlo físicamente. Esto permite su uso en pacientes con el síndrome *lock-in*.

  La modulación del ritmo sensoriomotor puede conseguirse tras una única sesión de entrenamiento. Posteriormente se requiere de cierto tiempo de aprendizaje máquina para generar un clasificador específico para el sujeto, que podrá utilizarse a continuación en sesiones con retroalimentación.

  Este es el modelo que se utiliza el BCI con el que se ha trabajado en este proyecto de fin de carrera.

Señales cerebrales adquiridas mediante Magnetoencefalograma (MEG-BCIs)

La corriente eléctrica producida por la actividad neuronal induce un campo magnético que puede ser registrado mediante magnetoencefalografías (MEG). El principal inconveniente de estos sistemas es que su uso se ve limitado al entorno del laboratorio (por lo menos actualmente), lo que limita las posibilidades de aplicación: un paciente no podría usarlo en su casa para controlar un entorno domotizado o para comunicarse con otras personas.

Por otra parte, comparado con las electroencefalografías, las magnetoencefalografías tienen una resolución espacial significativamente mejor, localiza la activación cortical de forma mucho más precisa. Además, la señal recogida tiene menos ruido ya que la MEG registra la actividad eléctrica primaria, cuyos campos magnéticos asociados no sufren problemas de atenuación, distorsión o modificación de la conductividad [18]

Los sistemas BCIs que emplean MEGs pueden servir para localizar el foco de actividad neuronal si con EEG no se obtuvieron resultados satisfactorios. Posteriormente, se puede tratar de utilizar un BCI que emplee EEG recolocando los electrodos y ajustando el rango de las frecuencias detectadas según la información recogida durante el uso del MEG-BCI.[19]

Señales cerebrales recogidas mediante resonancia magnética funcional (fMRI-BCIs)

Estos sistemas se basan en detectar las variaciones del nivel de oxígeno en sangre (abreviado BOLD response del inglés *Blood Oxygen Level Dependent Response*).

El incremento de la actividad neuronal va acompañada por un incremento local del metabolismo de la glucosa, incrementándose por tanto el consumo de glucosa y de oxígeno. Como consecuencia del consumo de glucosa realizado por el cerebro, las arterias craneales se dilatan permitiendo un aumento del flujo sanguíneo que provoca hiperoxigenación de los tejidos neuronales activos. La oxihemoglobina posee propiedades diamagnéticas, es decir, es repelida por la acción de un imán, mientras que la deoxihemoglobina tiene propiedades paramagnéticas, luego es ligeramente atraída por imanes. Se pueden aprovechar que la hemoglobina oxigenada (oxihemoglobina) y desoxigenada (desoxihemoglobina) tienen diferentes propiedades magnéticas para formar imágenes que representen las zonas activadas/desactivadas del cerebro. Así es como se obtienen las imágenes por resonancia magnética funcional (en inglés abreviado como fMRI, "functional magnetic resonance imaging").

Alcanza una resolución espacial del rango de milímetros y una localización

más precisa de la actividad neuronal que EEG, además permite detectar la activación de las zonas subcorticales, por debajo del córtex. Su uso de momento se ve restringido al laboratorio, pero se están realizando avances técnicos que pueden conducir a la creación de sistemas de resonancia magnética portátiles [20] en un futuro no muy lejano.

Recientemente se han realizado investigaciones utilizando fMRIs para que pacientes aprendan a regular la actividad neuronal de ciertas zonas del cerebro. De esta forma se ha conseguido que pacientes con dolor crónico varíen su percepción del dolor autorregulando la actividad de un área especifica del cerebro (el área rostral del córtex anterior cingulado). En general, se podría entrenar pacientes con enfermedades psicológicas a estimular áreas de su cerebro que se sabe que están implicadas en sus desórdenes, como es el caso de la psicopatía y la depresión y poder reducir así su sintomatología.

Owen y su equipo de investigadores lograron distinguir satisfactoriamente patrones de activación relativos a imaginación de movimiento (jugar al tenis) y a navegación espacial (a través de la casa del sujeto, comenzando por la puerta). Se pidió a una paciente que había sido diagnosticada de estado vegetativo persistente que realizara estas dos tareas cognitivas. Se comprobó que sus patrones de activación eran muy similares a los de sujetos sanos, y de esta forma se pudo demostrar que la paciente estaba realmente consciente y no en estado vegetativo.[21]

Señales cerebrales recogidas mediante topografía óptica. (NIRS-BCI)

La topografía óptica, o espectrografía del infrarrojo cercano, en inglés *Near Infrared Spectroscopy* (NIRS), también se basa en la detección de las variaciones del nivel de oxígeno en sangre (BOLD), pero explota las características ópticas en el rango de la luz visible y los infrarrojos cercanos de la oxihemoglobina y la desoxihemoglobina, en lugar de sus propiedades magnéticas como se hace con fMRI. La resolución espacial que se obtiene con la topografía óptica es comparable con la obtenida con las fMRIs, pero es mucho menos costosa y además es portable por lo que puede utilizarse en casa del paciente. Tiene la desventaja con respecto la resonancia magnética de que las señales recogidas son sólo del córtex, no del subcórtex.

Se ha demostrado que se puede detectar fácilmente mediante NIRS la activación cerebral debida a imaginar que se está realizando un movimiento (como “mover la mano derecha”). [5]

2.4.2. BCIs invasivos

Los métodos invasivos pueden medir la actividad neuronal de la superficie del córtex o del interior del mismo. Estos métodos tienen grandes ventajas respecto los métodos no invasivos en términos de calidad de la señal y dimensionalidad. Su principal desventaja es que requieren cirugía y los problemas a largo plazo plazo para mantener la estabilidad de los implantes y evitar infecciones.

Señales cerebrales adquiridas de la superficie del córtex. Electrocorticograma. (ECoG)

El electrocorticograma usa rejillas o tiras de electrodos que se insertan en la superficie del córtex (epidurales) o su interior (subdurales) que recogerán la actividad eléctrica del cerebro. Implica realizar una craneotomía al sujeto, aunque cuantos menos electrodos se requieran, menos invasiva será la cirugía a realizar (pueden introducirse tiras de electrodos en lugar de rejillas, a través de un pequeño agujero en el cráneo).

La principales ventajas de este sistema es la alta resolución espacial que se obtiene con respecto a los de los métodos no invasivos (decenas de milímetros vs centímetros), mayor ancho de banda de la señal (0-200 Hz vs 0-40Hz) y mayor amplitud (50-100 μVoltios vs 5-20 μVoltios) y menor vulnerabilidad ante artefactos eléctricos.

Tradicionalmente los ECoG se utilizan para localizar la "zona irritativa" en pacientes con epilepsia resistente a fármacos antes de que se sometan a una cirugía. Se han realizado extensos estudios sobre la viabilidad de usar ECoG para sistemas BCI en pacientes con epilepsia. La mayoría de estos estudios se realizaron analizando los datos recogidos offline, sin proporcionar *feedback* a los usuarios, obteniendo tasas de aciertos entre el 85 % y el 91 % al clasificar los datos obtenidos de un sujeto que extendia/desextendía el dedo medio (Scherer et al., 2003). En estudios en los que se ha utilizado retroalimentación se pedía a los pacientes que realizaran y/o imaginaran tareas motoras y lingüísticas (abrir y cerrar la mano, sacar la lengua, encogerse de hombros, decir la palabra "*move*"...) Se obtuvieron tasas de aciertos entre el 73 % y el 98 % al clasificar la imaginación de movimientos, después de un breve entrenamiento que duró entre 3 y 24 minutos. También se ha conseguido controlar el movimiento de un cursor imaginando eventos auditivos (canción favorita, voz, teléfono...) en lugar de eventos motores.

En un paciente de Esclerosis Lateral Amiotrófica (ELA) completamente paralizado se implantó una rejilla con 32 electrodos con el propósito de conseguir controlar un sistema BCI para comunicarse, pero no se alcanzó dicho

objetivo. Más de un año después de la cirugía aproximadamente el 50 % de los electrodos seguían recogiendo la actividad eléctrica de forma clara y estable. Es el único caso conocido en el que se ha implantado en una persona una rejilla de electrodos para su uso exclusivo como BCI.[5]

Señales cerebrales adquiridas dentro del córtex (intracorticales).

La adquisición de señales intracorticales puede realizarse con múltiples electrodos, con unos pocos o incluso con uno sólo. El electrodo debe estar muy próximo a la fuente de la señal y la matriz de electrodos debe ser estable durante un largo periodo de tiempo. Todavía deben mejorarse los las matrices de múltiples electrodos para su uso clínico, en animales se han conseguido utilizar manteniéndose estable durante dos años.

En experimentos realizados con monos, se comprobó que puede controlar un cursor y un brazo robótico usando los datos recogidos de unas pocas neuronas (18 células) del córtex motor. En un humano diagnosticado de tetraplegia tras dañarse la médula espinal se implantó una matriz de 96 electrodos en el córtex motor primario. Se demostró que los complejos de puntas, o *spike patterns* en inglés (un tipo de onda característica), son modulados con la intención de movimiento. El usuario fue capaz de controlar el televisor, abrir y cerrar una mano protésica incluso durante una conversación y otra serie de tareas. Después de 6.5 meses la sensibilidad de los electrodos se vio significativamente reducida. Este ha sido el primer ensayo piloto del implante de una matriz de electrodos intracorticales en un humano. [5]

2.5. Procesado de la señal

Normalmente, la señal obtenida del encéfalo mediante electroencefalografía se procesa antes de ser utilizada para generar un clasificador. Las trasformaciones más utilizadas son los filtros espaciales, la transformada de Fourier y los filtros de paso banda.

La señal adquirida mediante el sistema BCI empleado en este proyecto se representa indicando el valor del voltaje detectado por cada electrodo (canal) del casco en un determinado instante de tiempo t, generando 500 muestras de este tipo cada segundo. La señal detectada por un electrodo proviene de diferentes partes del cerebro. Esto es un problema si además la señal de interés es débil y existen otras fuentes que producen señales más fuertes en el mismo rango de frecuencia (como el ritmo sensoriomotor y el ritmo alfa cuando el sujeto está con los ojos abiertos).

Se utilizan diferentes técnicas de filtrado espacial, con el objetivo de

obtener para cada electrodo una señal más localizada. Los filtros espaciales también son de utilidad porque pueden reducir la dimensionalidad de la muestra (en lugar de tener un valor por cada canal, se pueden realizar una combinación de éstos y que el resultado sean unos pocos valores por muestra).

Un ejemplo de filtrado espacial es la aplicación de un filtro laplaciano, que básicamente consiste en restar a cada señal la media de los electrodos próximos al electrodo que captó dicha señal. La ecuación 2.1 muestra un ejemplo simple de filtro laplaciano. La elección del tamaño del vecindario a emplear determinará las características del filtro espacial. Normalmente se utilizan filtros laplacianos simples, como el del ejemplo, o filtros mayores empleando para calcular el nuevo valor de la señal de cada electrodo el vecindario que se encuentra a un 20 % de distancia de dicho electrodos, según el sistema internacional 10-20 de colocación de electrodos extendido (ver figura 2.9 de la sección 2.4.1.

$$C4_{LAP} = C4_{REF} - 1/4(C2_{REF} + C6_{REF} + FC4_{REF} + CP4_{REF}) \qquad (2.1)$$

Existen otros filtros espaciales, como los filtros de Análisis de Componente Principal (filtros PCA, del inglés *Principal Component Analysis*), que reducen la dimensionalidad del problema al tranformar las señales de los electrodos, previsiblemente correlacionadas entre sí, en un conjunto menor de señales no correlacionadas [22] . El filtro de Análisis de Componentes Independientes (filtros ICA del inglés *Independent Component Analysis*) trata de encontrar las supuestas fuentes independientes que contribuyen para obtener la señal de cada electrodo [23]. El filtro espacial de patrones comunes (CSP, del inglés *Common Spatial Patterns*) maximiza la varianza de señales de una clase a la vez que minimiza la de la otra, lo que lo hace adecuado para discriminar estados mentales caracterizados por efectos de sincronización/desincronización (efectos ERD/ERS, del ingles *Event Related Desynchronization/Event Related Synchronization*).[24]

Estos filtros y, en general, la mayoría de los filtros espaciales pueden expresarse como una transformación lineal de los datos originales. Por tanto se pueden implementar como el producto del vector original de datos adquiridos en un determinado instante por una matriz. Por ejemplo, el filtro laplaciano de la ecuación 2.1 podría expresarse de la siguiente forma (ecuación 2.5):

$$\begin{pmatrix} C4_{LAP} \\ C2_{LAP} \\ C6_{LAP} \\ FC4_{LAP} \\ CP4_{LAP} \\ \vdots \end{pmatrix} = \overbrace{\begin{pmatrix} 1 & -1/4 & -1/4 & -1/4 & -1/4 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix}}^{Matriz filtro} * \begin{pmatrix} C4_{REF} \\ C2_{REF} \\ C6_{REF} \\ FC4_{REF} \\ CP4_{REF} \\ \vdots \end{pmatrix} \tag{2.2}$$

Una vez filtrados los datos en el dominio del tiempo se pasan al dominio de la frecuencia, ya que es más fácil clasificar los datos que se encuentran expresados de esta forma. Muchas características distintivas de la señal se muestran más claras en el dominio de la frecuencia, lo que hace que sea más fácil discriminar la actividad del encéfalo realizada mientras se recogía determinada señal.[25]

La señal en el dominio de la frecuencia se representan mediante la Densidad Espectral de Potencia o PSD (del inglés*Power Spectral Density*), función matemática que informa sobre la distribución en frecuencia de la potencia de una señal. Desde el punto de vista matematico el PSD se define como la transformada de Fourier de la funcion de autocorrelacion $R(\tau)$. Esta función está definida para funciones estacionarias continuas. Dado que la señal con la que se trabaja no es ni estacionaria ni continua (aunque la señal propiamente dicha es continua sólo se tienen muestras del valor de la señal cada cierto tiempo, como si fuese una señal discreta), se calcula una estimación del PSD, no el PSD propiamente dicho. [26]

Para estimar el PSD se puede aplicar el método de Welch, que tiene la ventaja de no ser computacionalmente costoso al utilizar la transformada rápida de Fourier (FFT, del inglés *Fast Fourier Transform*). Además minimiza el ruido introducido por otros métodos en los que se basa.

El método de Welch, también llamado método del periodograma, divide la señal en varios bloques de datos consecutivos, que pueden solaparse o no, y calcula el periodograma de cada uno de estos bloques, posteriormente hace una media de los resultados obtenidos. [27]

El peoriodograma es otro método de estimación de del PSD basado el en cálculo de la transformada discreta de Fourier (DFT, del inglés *Discrete Fourier Transform*). Este método introduce ciertos problemas debido a que puede producir salidas abruptas. Para solucionarlo se puede multiplicar el resultado por un vector de valores conocidos como "ventanas", como la *ventana de Barret* o la *ventana Hamming.*[26] [27] [28]. El sistema BCI empleado en este proyecto de fin de carrera utiliza por defecto la *ventana de Hamming*

para suavizar los resultados obtenidos al calcular la DFT.[2]

Para calcular la DFT se utiliza el algoritmo de la Transformada Rápida de Fourier (FFT). La idea de este algoritmo es descomponer la transformada a calcular en otras más simples, una vez calculadas dichas transformadas, se agrupan y se reordenan para obtener el resultado final.[29]

El método de Welch para la estimación del PSD retorna un vector que representa las frecuencias desde 0 hasta la mitad de la frecuencia de muestreo (Fs/2) también llamada *frecuencia Nyquist*. La resolución de la señal en el dominio de la frecuencia será el inverso de la longitud del segmento utilizado en el método de Welch (1/L) multiplicado por Fs [28] [30]. Si, por ejemplo, la frecuencia de muestreo de la señal es Fs=500 Hz y el tamaño del segmento de datos empleado es L=250, se obtendría una resolución de 2 Hz. Esto quiere decir que se obtiene la estimación del PSD entre las frecuencias 0 a 250 Hz en bandas de frecuencia de 2 Hz.

Una vez se dispone de los datos del dominio de la frecuencia se aplica un filtro pasobanda u otro que se considere conveniente para seleccionar las bandas de frecuencia relevantes [25]. Por ejemplo, en el sistema BCI utilizado en este proyecto de fin de carrera aplica un filtro paso banda que descarta las frecuencias inferiores a 8 Hz y las superiores a 30 Hz.

Una vez procesados los datos se construye un clasificador que, previsiblemente, se comportará sustancialmente mejor que si se hubiese construido empleando los datos sin procesar.

Capítulo 3

Hardware y Software implicado

En este capítulo se comentan los elementos hardware y software que han sido utilizados en las sesiones de adquisición de datos realizadas y en el análisis posterior de los datos obtenidos.

3.1. Casco electroencefalógrafo y amplificador de señal. Programa Brain Vision Recorder.

Para captar las señales electroencefalógrafas del sujeto, se ha empleado un casco encefalógrafo modular Easy-Cap con electrodos de anilla pasivos, un amplificador V-amp de 8 canales de BrainProducts y el software Brain-Vision Recorder V-Amp edition de BrainProducts. Estos productos pueden adquirirse en España a través de la empresa Bionic Ibérica S.A., que se dedica a la importación, distribución y venta de aparatos de Electromedicina y de Investigación de alta tecnología para el sector Hospitalario y Universitario, así como para la industria farmacéutica [31].

Durante el desarrollo de este proyecto de fin de carrera estos materiales han sido proporcionados por el grupo de investigación de Computación Evolutiva y Redes Neuronales de la Universidad Carlos III de Madrid (EVANNAI) [1], cuyas principales líneas de investigación son la Computación Evolutiva y el Aprendizaje Automático. Desde este grupo se ha motivado el desarrollo de proyectos de fin de carrera sobre sistemas BCI, como el que realizó Javier Asensio "Desarrollo de una Interfaz Cerebro-Máquina" [2] y como éste donde se añaden nuevas funcionalidades a dicho BCI y se adquieren y analizan datos electroencefalográficos.

El casco encefalógrafo utilizado está fabricado por la empresa Easy Cap

GmbH, que colabora con la empresa Brain Products GmbH que es la que produce el amplificador V-Amp y el software BrainVision Recorder. El casco consta realmente de tres partes: un gorro de tela similar a un gorro de piscina, los soportes de los electrodos y los electrodos individuales. Este casco encefalógrafo tiene fijados de antemano los soportes de los electrodos en las posiciones correspondientes que se muestran en la figura 3.1 [33], siguiendo el sistema internacional 10-20 de colocación de los electrodos, explicado en la sección 2.4.1. Se muestra una foto del casco con sus accesorios en la figura 3.2. En estos soportes se fijan los electrodos cada vez que se va realizar una sesión de adquisición. No se utilizan todas las posiciones del casco, ya que el amplificador V-Amp del que se dispone es de ocho canales.

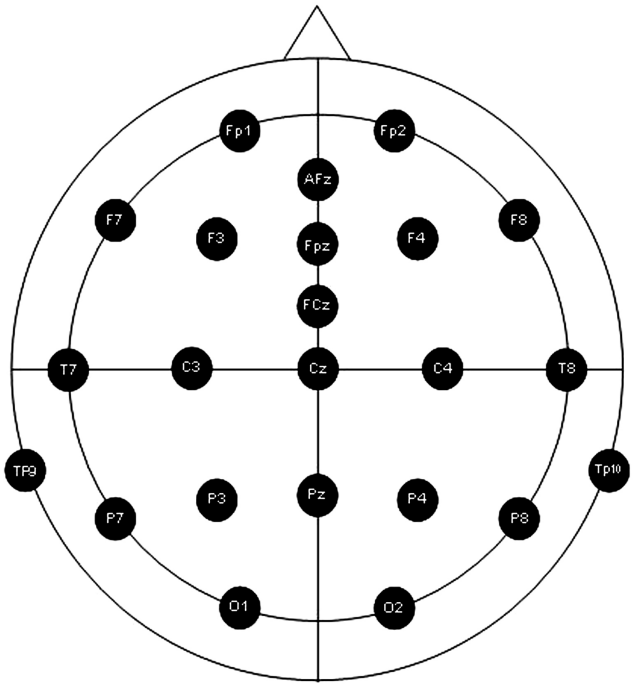

Figura 3.1: Posiciones de los soportes de los electrodos en el casco encefalógrafo Easy-Cap de 19/21 canales

El amplificador V-Amp de ocho canales dispone de una interfaz gráfica que se muestra en la pantalla TFT del propio dispositivo y se conecta al ordenador mediante un puerto USB. A él van a parar diez electrodos, los ocho correspondientes a los canales de los que se van a extraer datos y dos adicionales, el de tierra y el de referencia. En la figura 3.3 se muestra una

Figura 3.2: Casco encefalógrafo con los accesorios requeridos para realizar una sesión de adquisición.

foto de un dispositivo similar al utilizado pero con entradas para 16 canales en lugar de para ocho. Las posiciones del casco que se han utilizado durante las sesiones de adquisición realizadas en este proyecto de fin de carrera son F3, F4, FCz, Cz, C4, T8, C3 y T7 y además AFz y Fcz como electrodos de tierra y de referencia respectivamente. En la tabla 5.4 de la sección 5.2.4, en el apartado "Sobre la colocación de los electrodos"se muestra en qué canal del amplificador se conectó cada electrodo.

Figura 3.3: Amplificador de señales neurológicas Brain Products V-Amp de 16 canales

Mediante el programa BrainVision Recorder se controlan ciertas funciones del amplificador, como es la posibilidad de generar una señal aleatoria de test para poder hacer pruebas sin tener la necesidad de que un sujeto se coloque el casco. Además, permite supervisar el nivel de impedancia de cada canal de una forma cómoda e intuitiva mediante un mapa que muestra cada uno de los electrodos en su posición con un código de color que nos indica el nivel de impedancia (Figura 3.4). Mediante este programa se pueden grabar los

datos adquiridos en un fichero, o también se pueden enviar vía TCP/IP a través del módulo de *Acceso Remoto de Datos*, enviando los datos al puerto que se desee. Esta funcionalidad es la que más importancia tiene para el desarrollo del proyecto, ya que es la que hace que el programa MindReader pueda recibir los datos del casco en tiempo real.

Para poder utilizar el programa BrainVision Recorder y el amplificador V-Amp se requiere además de una llave hardware USB Brain Products, que es un dispositivo anti-copia que debe estar conectado vía USB al ordenador para permitir utilizar los programas de Brain Products.

3.1.1. Colocación del casco y electrodos

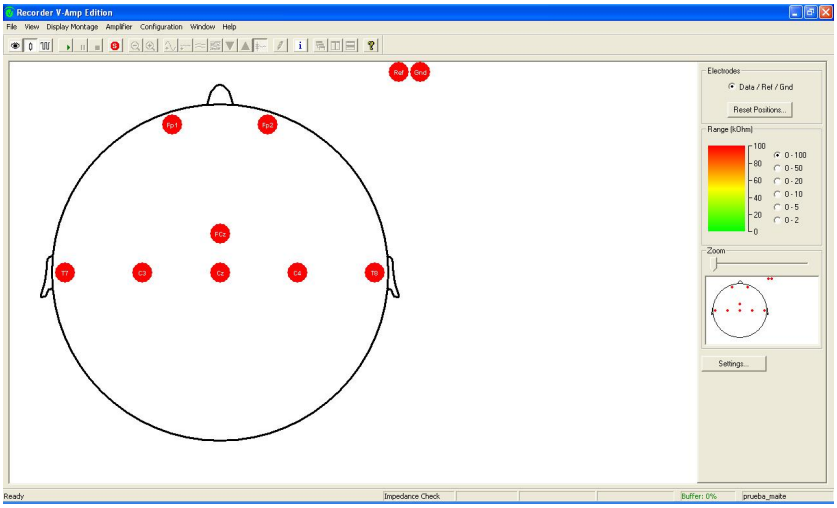

Figura 3.4: Pantalla de control de impedancia del programa BrainVision Recorder

El primer paso para realizar una sesión de adquisición es colocar el casco al sujeto y reducir la impedancia de los canales utilizados para obtener una buena calidad de señal. Teóricamente este proceso de colocación del casco suele durar entre 20 y 30 minutos. En las sesiones de adquisición de datos realizadas a lo largo de este proyecto de fin de carrera se necesitó mucho más tiempo, en torno a una hora, y en una ocasión hasta dos horas; aunque también se consiguió en otra sesión realizarlo en algo menos de cuarenta minutos. Es un proceso algo pesado tanto para el sujeto como para los colaboradores que ayudan a reducir las impedancias de los electrodos. En la figura 3.5 se muestra a uno de los sujetos que realizaron sesiones de adquisición en medio del proceso de reducción de impedancias.

Figura 3.5: Reduciendo impedancias durante la tercera sesión de adquisición de datos de David

A la hora de colocar el casco se comienza por la frente, hay que asegurarse de que el electrodo Cz está centrado y que la posición de los electrodos laterales es simétrica. Las orejas del sujeto deberían salir por los agujeros correspondientes para permitir que el casco quede bien fijado. El casco se fija con el arnés torácico, que se fija al gorro mediante los dos cierres de tipo 'clic', debe colocarse por debajo de los hombros, con las cintas del casco cruzadas a través (aunque si esto le resulta molesto al sujeto puede colocarse sin cruzar las cintas). La tensión de las cintas puede regularse empleando las distintas posiciones del arnés para que el sujeto se sienta más cómodo.

Una vez fijado el casco en la cabeza del sujeto, los electrodos se sitúan en sus adaptadores. Hay que colocar el electrodo de forma que la unión del cable con la arandela quede en la parte más estrecha del soporte, con cuidado de no doblar ni romper el cable. En realidad, también se podría realizar esto antes de que fijar el casco en la cabeza del sujeto, de forma que puede dejarse preparado el casco con los electrodos antes de que llegue el sujeto y así ganar algo de tiempo. El otro extremo de cada uno de los electrodos se conectará al amplificador V-Amp.

Cuando el casco ya está fijo y con los electrodos conectados al amplificador, se pasa a reducir impedancias de cada canal, para lo que se requiere la aplicación BrainVision Recorder (deberá conectase la llave hardware anti-

copia). Hay que indicar al programa qué electrodo se corresponde con cada canal del amplificador V-amp, esto se hace en el menú *Display Montage >Edit*, y la configuración puede guardarse para reutilizarse en sucesivas sesiones. Mediante la pantalla del programa BrainVision Recorder de control de impedancia, a la que se accede pulsando el botón *impedancia*, se comprobarán las impedancias de cada uno de los electrodos, hasta obtener un valor de impedancia cercano a los 2 kilo-ohmios, suficiente para obtener una señal clara. La interfaz de control de impedancia de este programa tiene el aspecto que se muestra en la figura 3.4, en ella se representa una cabeza con los electrodos colocados en las posiciones que se indicaron anteriormente. Los electrodos se representan inicialmente en rojo y según la impedancia se reduce pasará por una escala de colores (naranja, amarillo, amarillo verdoso...) hasta alcanzar el color verde. En la parte derecha se puede seleccionar la escala que se utiliza para medir la impedancia. Conviene seleccionar al principio la escala más alta (de 0 a 100 Kilo-ohmios) y cuando se ha conseguido un nivel verde en esa escala, bajar a la siguiente hasta conseguir lo mismo, y así sucesivamente hasta que se consiga un color verde en la escala de 0 a 5 Kilo-ohmios y amarillento en la escala de 0 a 2 Kilo Ohmios. A menor impedancia, mayor calidad de señal se obtendrá.

Además se necesitan los materiales siguientes:

- Varias torundas (bastoncillos limpiadores)
- Alcohol al 70 %
- Abrayt 2000 (Gel electrolito abrasivo)
- Una jeringuilla sin aguja para rellenar con el gel
- Toallitas o toalla para limpiar el gel sobrante
- Cepillo de dientes para limpiar el casco

Si es posible, el sujeto debería tener la cabeza lavada y sin haber utilizado aditivos como acondicionadores, lacas o geles fijadores. Esto ayuda significativamente a obtener impedancias bajas. Con la parte del bastoncillo sin algodón, hay que retirar el pelo hacia los lados hasta que sea visible el cuero cabelludo. Es importante comprobar que el adaptador no está situado encima de un lunar, cicatriz o grano, ya que la aplicación del gel abrasivo podría causar inflamaciones. Si es necesario, se deberá desplazar el soporte del electrodo hacia un lado.

A continuación, se remoja la parte del algodón de la torunda en el alcohol y se desengrasa el cuero cabelludo a través del orificio del electrodo,

introduciendo el bastoncillo por el agujero y moviéndolo de forma circular o describiendo una cruz. Utilizando solo alcohol se consiguen impedancias entre los 10 y los 20 Kilo-ohmios. Para conseguir impedancias más bajas se debe mojar el algodón de otro bastoncillo con el gel abrasivo y hacer el mismo movimiento. Es muy importante en este paso no ejercer demasiada presión, ya que en caso contrario puede aparecer caspa, e incluso lesionar el cuero cabelludo del sujeto. Para evitar lo anterior, conviene además cambiar frecuentemente de torunda, ya que el algodón va perdiendo su forma y consistencia y al final la torunda queda reducida al palo de madera. Hay que insistir al sujeto en que comunique si nota dolor, aunque sea leve, ya que en teoría este proceso no debería ser doloroso. Una vez alcanzado el nivel de impedancia deseado, hay que rellenar el espacio de la piel al electrodo con gel electrolito con ayuda de una jeringa.

Este proceso deberá repetirse hasta que todos los electrodos tengan impedancias bajas. El electrodo de Tierra y el de Referencia deberían colocarse al principio y ambos con impedancias muy bajas. Esto facilitará alcanzar buenas impedancias en el resto.

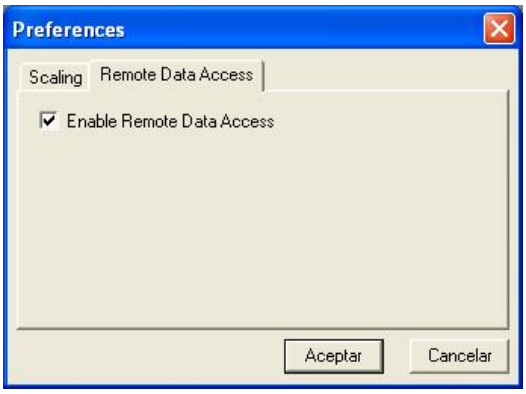

Figura 3.6: BrainVision Recorder: Activar funcionalidad de Acceso Remoto a Datos

Por último, se debe activar el servidor Remote Data Access del programa BrainVision Recorder para tener acceso a los datos que está recogiendo en tiempo real desde la aplicación MindReader. Esto se realiza desde la opción *Remote Data Access* del menú *Configuración > Preferences* (ver figura 3.6). Una vez activada esta funcionalidad, el programa MindReader (o cualquier otro preparado para ello) podrá conectarse mediante una conexión TCP/IP a los puertos 51234 para obtener datos de precisión de 2^{16} ó 51244 para obtener datos de precisión de 2^{32}.

3.1.2. Mantenimiento del casco

Una vez terminada la sesión de adquisición hay que sacar los electrodos de los adaptadores, con mucho cuidado una vez más de no romper ni doblar los cables. Es conveniente ayudarse de un bolígrafo o bastoncillo para evitar tirar de la soldadura que une la arandela al cable. Además hay que limpiar el casco y los electrodos con agua hasta eliminar todos los restos de gel abrasivo, para ello podemos ayudarnos del cepillo de dientes. Se deben secar los elementos a temperatura ambiente para guardarlos de nuevo perfectamente limpios y secos.

3.2. MindReader

El programa MindReader ha sido utilizado en este proyecto de fin de carrera para adquirir y procesar datos electroencefalógrafos y además se han introducido ciertas modificaciones en él, añadiéndole nuevas funcionalidades.

MindReader y nnt fueron desarrolladas por Javier Asensio en su PFC "Diseño de una Interfaz Cerebro-Máquina". El objetivo principal de su proyecto era «el diseño de una plataforma que abarque todos los aspectos, desde la recogida y almacenamiento de la señal, pasando por el entrenamiento del clasificador y su uso en tiempo real. Además, se pretende desarrollar una pequeña aplicación que permita el control mental de un apuntador. Lo que se persigue pues, es un sistema funcional, que recoja las señales producidas por el cerebro de un sujeto, las procese, las etiquete, sea capaz de realizar una asociación entre lo que piensa el sujeto y una acción, y la ejecute debidamente en un entorno diseñado para tal efecto. Se quiere que la plataforma sea lo más flexible posible para poder ampliarla y construir otras aplicaciones a partir de esta.» [2]. Esta plataforma requiere de elementos hardware, como son el casco encefalógrafo y el amplificador de señal y de elementos software, como son el programa MindReader y el programa nnt.

MindReader es la aplicación principal del sistema, puede recibir las señales electroencefalógrafas en tiempo real enviadas por el programa BrainVision Recorder (que a su vez las recibe del amplificador), permite realizar sesiones de adquisición de datos, sesiones de adquisición de datos con retroalimentación, y comprobar el funcionamiento de los clasificadores dirigiendo el apuntador de un ratón por la pantalla indicando el movimiento a seguir mediante diferentes procesos mentales.

Cuando se realiza una sesión de adquisición de datos se genera un fichero `*_time.raw` con los datos en el dominio del tiempo, estos datos están etiquetados a través del fichero `*_patterns.mrp` que indica a qué clase pertenece

cada uno. Además de adquirir datos y clasificarlos, se procesan los datos y se genera un fichero `*_psd.raw` con los datos en el dominio de la frecuencia, filtrando ciertas frecuencias máximas y mínimas que se indican al programa por parámetros. En las sesiones de adquisición pueden utilizarse señales auditivas o visuales para indicar al sujeto que debe cambiar de proceso mental.

El programa MindReader permite realizar sesiones de adquisición de datos con retroalimentación, para ello se utilizará un clasificador entrenado previamente. La retroalimentación consiste en que además de mostrarle al sujeto en qué debe pensar en determinado momento, se muestra la salida que se obtiene del clasificador en tiempo real según las señales electroencefalógrafas que emite el sujeto. De esta forma no es sólo la máquina la que está aprendiendo a clasificar los pensamientos del sujeto, si no que el propio sujeto aprende también a pensar de forma que la máquina pueda clasificar sus pensamientos.

Además MindReader proporciona una aplicación que permite utilizar un clasificador entrenado para dirigir el apuntador de un ratón por la pantalla.

3.3. Programa nnt

En este proyecto se ha utilizado el programa nnt. Este programa permite generar y entrenar redes de neuronas a partir de los datos de uno o varios ficheros mediante una interfaz de consola. Utiliza la librería de código libre FANN (*Fast Artificial Neural Network*). A través de un fichero de configuración, cuyo nombre se pasa como argumento en la línea de comandos al invocar el programa, se pasan los parámetros de configuración de la red de neuronas (arquitectura de la red, número de neuronas por capa) y del entrenamiento (momento y tasa de aprendizaje, los ficheros donde se encuentran los datos, porcentaje de datos a emplear en el entrenamiento, número de ciclos de entrenamiento a realizar) así como otros requeridos por la aplicación (el nombre del archivo de salida a generar, cada cuántos ciclos se generará un informe de errores...). Para ejecutar esta funcionalidad hay que invocar el programa como se indica a continuación:

```
$ nnt net <config_file_path>
```

Además nnt genera un archivo `*.arff` de los datos que se encuentren en un fichero, para que puedan ser procesados por el programa WEKA. En este caso, del fichero de configuración que se le pasa como parámetro sólo se tienen en cuenta los parámetros *porcentaje de entrenamiento*, y los nombres

de los ficheros de entrada y de salida. Para ejecutar esta funcionalidad hay que invocar el programa como se indica a continuación:

```
$ nnt arff <config_file_path>
```

Es importante seguir la sintaxis adecuada a la hora de escribir el archivo de configuración, ya que si el nombre de los parámetros es incorrecto la aplicación puede tener una salida inesperada. En la figura 3.7 se muestra un ejemplo de archivo de configuración válido, que cargaría los datos de los periodos p3 a p9 de la tercera sesión de adquisición realizada por David. Nótese que cada línea del fichero es una pareja clave-valor, el *valor* debe ser del tipo adecuado al parámetro que indica la *clave*. Los nombres de las claves de cada parámetro están especificados en el archivo `ConfigConst.h` del código fuente, y deben seguir la sintaxis indicada o el programa no será capaz de determinar el parámetro al que se quiere dar valor. Los nombres de los ficheros de entrada deben ir seguidos por un caracter ';' , excepto el último, para separarlos entre sí. Hay que tener en cuenta que no se escribe el nombre completo del archivo, sino el prefijo común de los archivos `*_psd.raw` y `*_patterns.mrp` que se corresponde con un conjunto de datos.

```
1 RUN_ID david_s3_p3-p9
2 OUTPUT_PATH C:\Maite\Proyecto\Mindreader_1.0\sesiones\sesion_3\
      nnt_out\
3 ARCH 20
4 LEARN_RATE 0.15
5 LEARN_MOMENTUM 0.8
6 TRAIN_PERC 20
7 FILE_PREFIX C:\Maite\Proyecto\Mindreader_1.0\sesiones\sesion_3\
      s3david03;C:\Maite\Proyecto\Mindreader_1.0\sesiones\sesion_3\
      s3david04;C:\Maite\Proyecto\Mindreader_1.0\sesiones\sesion_3\
      s3david05;C:\Maite\Proyecto\Mindreader_1.0\sesiones\sesion_3\
      s3david06;C:\Maite\Proyecto\Mindreader_1.0\sesiones\sesion_3\
      s3david07;C:\Maite\Proyecto\Mindreader_1.0\sesiones\sesion_3\
      s3david08;C:\Maite\Proyecto\Mindreader_1.0\sesiones\sesion_3\
      s3david09
8 CYCLES 200
9 ERROR_REPORT 10
```

Figura 3.7: Ejemplo de archivo de configuración válido para el programa nnt

Como se ha comentado anteriormente, el programa nnt es una aplicación auxiliar desarrollada en el contexto del proyecto de fin de carrera de Javier Asensio "Diseño de una Interfaz Cerebro-Máquina"[2].

En [2] se encuentran explicaciones más detalladas del uso y estructura de estos programas.

3.4. WEKA

El programa WEKA[35] está implementado en Java y consiste en una recopilación de algoritmos de aprendizaje automático para afrontar tareas de minería de datos. Los algoritmos pueden aplicarse directamente a un conjunto de datos a través de una interfaz gráfica bastante amigable o se pueden invocar directamente desde el código de cualquier aplicación implementada en Java. WEKA dispone de herramientas para preprocesado de datos, clasificación, regresión, clustering, reglas de asociación y visualización. Su código fuente es libre, publicado bajo la licencia GNU General Public License[36]. Por esta razón es muy apropiado para su uso en docencia y para desarrollar nuevos algoritmos de aprendizaje automático.

En este proyecto de fin de carrera el programa WEKA se ha empleado para analizar los datos obtenidos del programa MindReader durante las sesiones de adquisición, entrenando diferentes tipos de clasificadores. Tanto el programa nnt como el programa MindReader desde la nueva interfaz de *entrenamiento de redes* permiten entrenar una red de neuronas de tipo perceptrón multicapa (MLP) o cualquier otro clasificador que se haya implementado. No obstante estos programas tienen varios defectos, como es que no se pueden especificar qué datos se van a utilizar para formar el conjunto de entrenamiento y cuáles para el conjunto de test. Sólo permiten indicar uno o varios ficheros y el porcentaje de los datos que se va a utilizar para entrenamiento, y los datos de estos ficheros se seleccionan pseudoaleatoriamente hasta que los conjuntos de test y entrenamiento alcanzan el tamaño deseado. Además los procesos de lectura y escritura de estos programas no son muy eficientes comparado con WEKA. La aplicación WEKA ha sido muy útil ya que permite entrenar diferentes tipos de clasificadores de forma eficiente, pudiendo elegir ficheros diferentes para el conjunto de entrenamiento y el de validación a través de una interfaz que permite cambiar cómodamente los parámetros de los clasificadores y del entrenamiento a realizar.

3.5. Octave

GNU Octave [37] es un lenguaje de alto nivel principalmente destinado a realizar cálculos numéricos. Dispone de una interfaz de línea de comandos desde la que se pueden resolver tanto problemas lineales como no lineales,

o realizar otros experimentos numéricos. En su mayor parte es compatible con Matlab [38], teniendo una gran ventaja sobre éste: GNU Octave es de libre redistribución, y su licencia se rige bajo los términos de la GNU General Public License[36]. Por esta razón es muy indicado su uso en docencia.

Octave se ha utilizado para implementar y ejecutar un script que genera un archivo `*.arff` con los datos en el dominio de la frecuencia que puede ser procesado por el programa WEKA. Los datos en el dominio de la frecuencia son calculados a partir de los datos de los ficheros _time.raw que contienen los datos en el dominio del tiempo obtenidos por el programa MindReader. El programa nnt al hacer esta transformación normaliza los datos del dominio de la frecuencia y además los desordena, mientras que esto no ocurre con este script. Esto ha permitido procesar los datos con WEKA y probar si se obtenían resultados distintos al aleatorizar y/o normalizar los datos antes de utilizarlos para entrenar un clasificador.

Capítulo 4

Ampliación de la herramienta

4.1. Punto de partida: primera versión del programa MindReader

En esta sección se van a comentar las características de la primera versión del programa MindReader. Se enumeran sus funcionalidades, los ficheros de salida y cómo se obtienen mediante los procesadores que se aplican a los datos adquiridos, así como la dependencia del uso de otras aplicaciones externas (nnt u Octave) a la hora de obtener los clasificadores utilizados en MindReader. Por último se comentan los parámetros que se utilizan tanto en la aplicación MindReader como en nnt, y se analiza la forma de cargar los datos de la aplicación nnt.

4.1.1. Funcionalidades de la primera versión del programa MindReader

En la sección 3.2 se hizo una breve descripción del programa MindReader y sus funcionalidades. La versión inicial del programa MindReader, desarrollada en el contexto del proyecto de fin de carrera de Javier Asensio [2], proporciona las siguientes funcionalidades:

1. Conexión con un servidor de datos (como *Brain Vision Recorder*) especificando su dirección IP y su puerto.
2. Monitorización de la señal recibida por cada canal.
3. Realización de sesiones de adquisición de datos mediante interfaz visual.
4. Realización de sesiones de adquisición de datos mediante interfaz auditiva.

5. Realización de sesiones de adquisición de datos con retroalimentación mediante interfaz visual.

6. Realización de sesiones de adquisición de datos con retroalimentación mediante interfaz auditiva.

7. Aplicación para controlar el dispositivo apuntador mediante un clasificador obtenido a través de la interfaz visual.

8. Aplicación para controlar el dispositivo apuntador mediante un clasificador obtenido a través de la interfaz auditiva.

4.1.2. Ficheros de salida

Durante las sesiones de adquisición de datos, ya sean con retroalimentación o no, se generan tres ficheros de salida cuyo nombre está compuesto por el prefijo que se haya especificado en la casilla contigua al texto "Fichero para guardar la sesión" y los sufijos `_patterns.mrp`, `_psd.raw` y `_time.raw`. El primero contiene la información sobre la sesión (qué parámetros se han utilizado) y en qué línea de los otros dos ficheros comienza cada bloque de datos perteneciente a determinada clase.

En el fichero de sufijo `_time.raw` existe una línea por cada instante de tiempo recogido con los datos del valor de la señal en dicho momento. La frecuencia de muestreo es de 500 Hz, luego 500 líneas se corresponden con los datos recogidos en un segundo. Estos datos se procesan calculando el PSD mediante el método de Welch, y filtrando el PSD obtenido de forma que sólo se toman las frecuencias comprendidas entre la frecuencia mínima y la frecuencia máxima especificadas. Cada línea del fichero `_psd.raw` se corresponde con el cálculo de la estimación del PSD para el instante cuyos valores están recogidos en esa misma línea pero del fichero `_time.raw`, que contiene los datos en el dominio del tiempo. Esta estimación se calcula a partir del segmento de señal de longitud especificada por el usuario (*número de muestras para la FFT*), formado por esa línea del fichero `_time.raw` y las *número de muestras para la FFT* muestras anteriores.

Cuando se realiza una sesión de adquisición de datos con retroalimentación, además de estos tres ficheros, se genera otro con el sufijo `_classifier_results.raw`. Cada vez que se muestra al usuario el resultado de la clasificación se escribe una línea en este fichero con dos valores. El primero es el identificador numérico del proceso mental que se indica al usuario en dicho instante de tiempo, y el segundo es la salida del clasificador al presentarle la señal adquirida. La filas del fichero cuyos dos valores coinci-

dan representan aciertos del clasificador, cuando son distintos el clasificador ha clasificado incorrectamente la señal.

4.1.3. Procesadores de datos. *DataProcessorLink* y *DataProcessor*

Los ficheros de salida y el procesado de la señal en el dominio del tiempo que realiza el programa MindReader viene determinado por los procesadores que se han definido y especificado que se van a utilizar en el código fuente.

La figura 4.1 representa el diagrama de clases de los procesadores implementados para la aplicación MindReader. Implementan un patrón composite (ver figura 4.2), asumiendo la clase `DataProcessor` el rol de *Component* y la clase `DataProcessorLink` el rol de *Composite*.

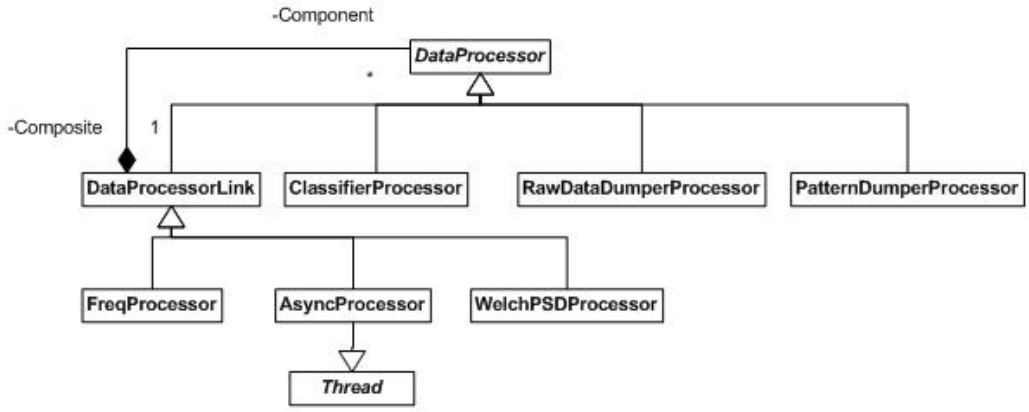

Figura 4.1: Diagrama de clases: DataProcessor e hijos

La clase `DataProcessor` sirve para implementar un procesador que realiza determinada acción sobre conjuntos de datos que se le irán pasando sucesivamente. Si la salida del procesador se va a utilizar como entrada de otro procesador entonces el procesador deberá implementarse a través de una clase que herede de la clase `DataProcessorLink`.

Los procesadores de tipo `DataProcessorLink` y los que heredan de él disponen de un método `DataProcessorLink::addOffspring(DataProcessor *dp)` que permite que se le añadan "procesadores descendientes", cuya entrada será la salida del procesador padre. El procesador padre mantiene una referencia a sus procesadores hijos que le permite comunicarse con ellos.

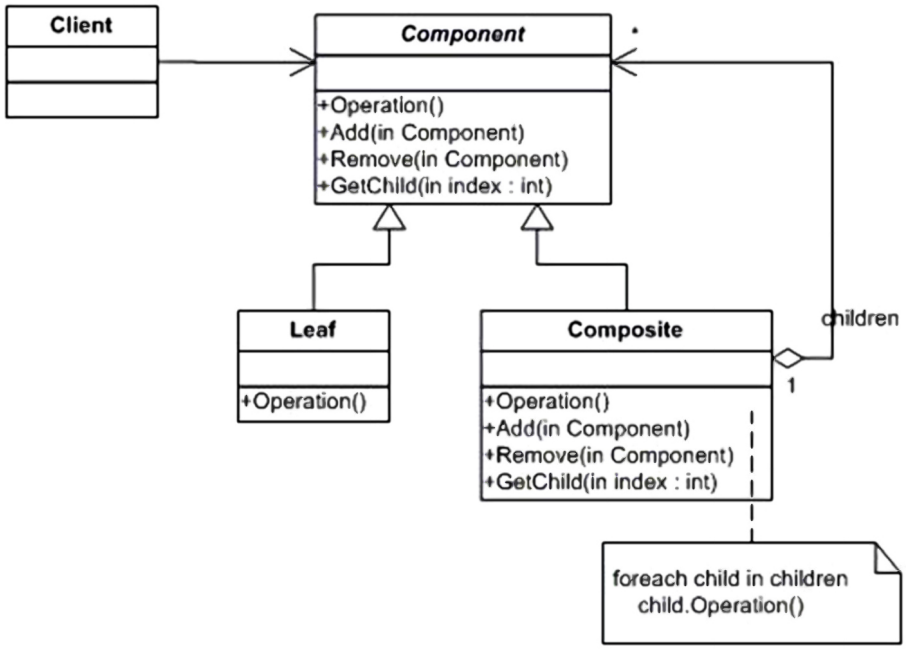

Figura 4.2: Diagrama de clases: Patrón de diseño composite

Por otra parte, las clases que heredan de la clase `DataProcessor` (y por tanto también `DataProcessorLink` y sus clases hijas) deben implementar los métodos `initSession`, `processData` y `endSession`. En el método `initSession` se deben realizar todas las operaciones preparatorias para que el procesador realice la acción para la que ha sido diseñado que no requieran de los datos que se vayan a procesar. Este método se invoca al inicio de la sesión de procesamiento de datos que se vaya a realizar, si el procesador es de tipo `DataProcessorLink` una vez realizadas las operaciones de inicialización invocará a los métodos `initSession` de sus procesadores descendientes.

Una vez hecho esto se puede llamar al método `processData`, que recibe un objeto de tipo `DataInstant`, tantas veces como sea necesario (por ejemplo, mientras se sigan recibiendo datos encefalográficos del servidor de datos). En cada llamada se realizará la operación sobre el conjunto de datos representado por el objeto `DataInstant`. Si el procesador es de tipo `DataProcessorLink` se invocará a continuación a sus procesadores hijos, pasándoles un nuevo objeto `DataInstant` que contiene el resultado de aplicar la operación a los

datos contenidos en el objeto `DataInstant` que recibió el procesador padre.

Cuando ya no se van a procesar más datos se invoca al método `endSession` que concluye la operación y/o libera recursos. En el caso de las clases que heredan de `DataProcessorLink`, este método se encarga de llamar a los métodos análogos de sus procesadores descendientes.

En la figura 4.3 se muestra las llamadas que se hacen durante una sesión de adquisición de datos a los procesadores para obtener la salida comentada anteriormente en la sección 4.1.2. Se utilizan dos procesadores de tipo `RawDataDumperProcessor` que se encargan de volcar los datos a los ficheros correspondientes, un procesador de tipo `PatternDumperProcessor` que se encarga de generar el fichero `*_pattern.mrp`, un procesador de tipo `WelchPSDProcessor` que se encarga de realizar la estimación del PSD y un procesador de tipo `FreqProcessor` que se encarga de filtrar las frecuencias máximas y mínimas. El procesado se produce en dos hilos de distinta prioridad gracias a la utilización de un objeto de la clase `AsyncProcessor`. Se va a explicar brevemente los aspectos más relevantes de la implementación de estos procesadores y su uso mientras se está realizando una sesión de adquisición de datos.

Volviendo al diagrama de secuencia de la figura 4.3, el objeto `dP` de tipo `DataProcessorLink` actúa como soporte de los procesadores implicados en la obtención de datos en una sesión de adquisición. Al inicializarse se inicializan todos sus procesadores descendientes (y los descendientes de éstos) en cadena. El objeto `dP` tiene tres descendientes directos: un objeto llamado `rdap` de tipo `RawDataProcessor`, un objeto llamado `ddp` de tipo `PatternDumperProcessor` y un objeto llamado `wp` de tipo `WelchPSDProcessor`. El objeto `dP` es inicializado justo antes de que se comiencen a recibir datos en el servidor, momento en el que se invoca sucesivamente a su método `processData` pásandole los datos recibidos en cada instante según la frecuencia de muestreo definida.

En el método `RawDataDumperProcessor::initSession` se abre en modo lectura el fichero cuyo nombre se ha recibido por parámetros, en el caso del objeto `rdap` será el fichero `*_time.raw`. Cada vez que se invoca al método `RawDataDumperProcessor::processData` se escribe una línea en dicho fichero con los datos contenidos en el `DataInstant` recibido, precedidos por el número de dato que se está procesando (que además coincidirá con el número de línea del fichero).

El método `PatternDumperProcessor::initSession` se encarga de abrir en modo lectura el fichero cuyo nombre se le pasa por parámetros, en el caso del objeto `ddp` será el archivo `*_patterns.mrp`. Además escribe la cabecera del fichero con los parámetros que se están utilizando durante la sesión de adquisición de datos.

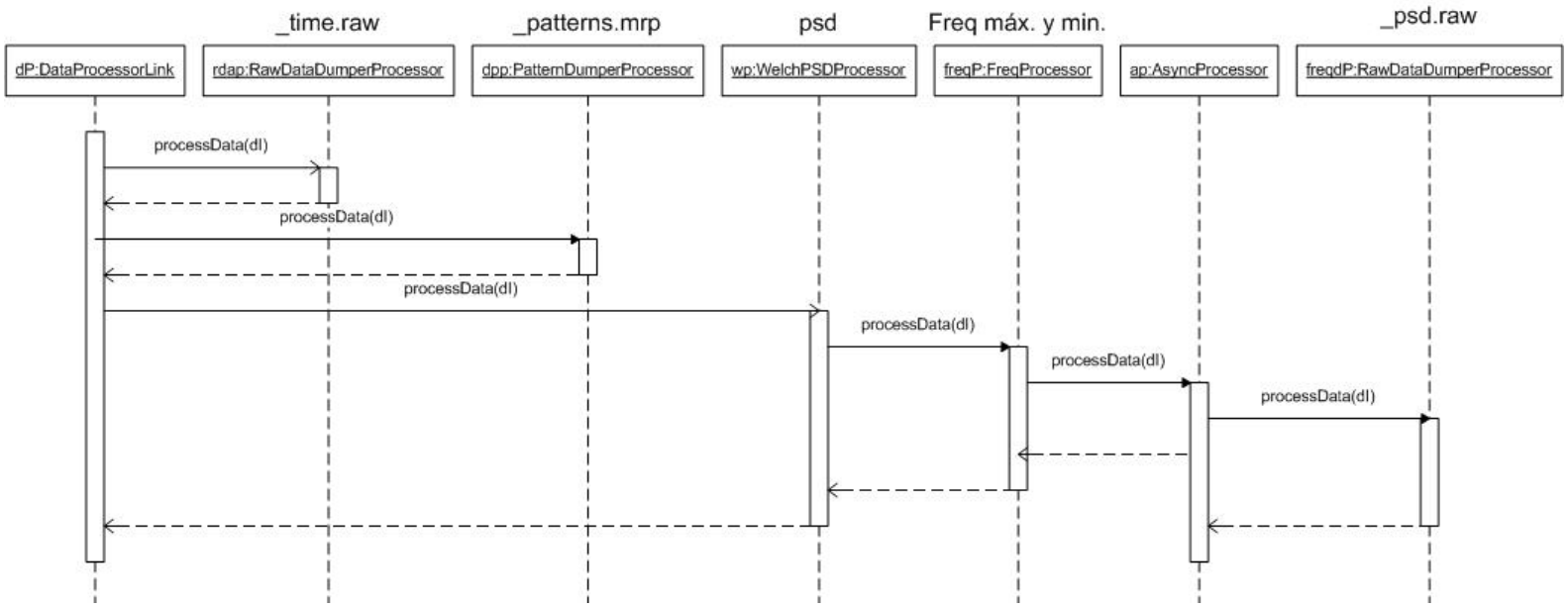

Figura 4.3: Diagrama de secuencia: Procesamiento de datos (I)

Posteriormente, cada vez que se invoca al método `PatternDumperProcessor::processData` se comprueba, mediante el objeto de tipo `DataSession` que comparten todos las clases descendientes de `DataProcessor`, la clase que se está mostrando al usuario en dicho momento, y si es distinta de la que se estaba mostrando la vez anterior en que se invocó al método, se escribe una línea en el fichero indicando el número de dato procesado (que coincide con el número de línea dentro del fichero `*_time.raw` donde figurarán los datos que se están procesando en dicho instante) y el identificador numérico de la clase que se muestra al usuario.

El objeto `wp` se encarga de aplicar el método de Welch para calcular la estimación del PSD. Tiene un procesador descendiente, un objeto de tipo `FreqProcessor` llamado `freqP`. Este procesador a su vez tiene otro descendiente, un procesador de tipo `AsyncProcessor` cuyo descendiente es otro procesador de tipo `RawDataDumperProcessor` llamado `freqdP`.

El método `WelchPSDProcessor::initSession` se encarga de reservar memoria para un vector de objetos de tipo `DataInstant` cuyo tamaño será el de la ventana que se va a utilizar a la hora de aplicar el método de Welch. Según se va invocando al método `WelchPSDProcessor::processData` este vector se va rellenando introduciendo el `DataInstant` recibido según el algoritmo FIFO y se estima el PSD, generando un `DataInstant` del tamaño adecuado que contiene la señal en el dominio de la frecuencia. Este resultado se pasará al método `FreqProcessor::processData` de su procesador hijo `freqP` que eliminará las frecuencias inferiores y superiores a las frecuencias mínima y máxima respectivamente y pasará un objeto `DataInstant` con dicho contenido al método `AsyncProcessor::processData` del objeto `ap`. El uso de procesadores asíncronos hace que el procesamiento de sus procesadores descendientes (en este caso el procesador `freqdP`, que se encarga de volcar el contenido del fichero `*_psd.raw`), no se produzca en el preciso momento en que se invoca al método `processData`, sino más adelante cuando la CPU esté más desahogada de trabajo. De esta forma la escritura del segundo fichero puede posponerse y las restricciones de tiempo real del sistema no se ven comprometidas. Si no se usase este procesador asíncrono, la aplicación no tendría tiempo aplicar todos los procesadores antes de recibir el siguiente conjunto de datos del servidor y podrían etiquetarse las muestras incorrectamente.

El diseño del sistema, al utilizar el patrón Composite, hace que sea sencillo añadir un nuevo elemento al procesado de datos, implementando un procesador que herede de `DataProcessorLink` si su salida va a ser la entrada de otro procesador, o de `DataProcessor` en caso contrario. Además, si se utiliza el procesador `AsyncProcessor` su/sus procesador/es descendiente/s

se ejecutará/n en otro hilo, lo que permite no sobrecargar el hilo principal de la aplicación y que se mantengan las restricciones de tiempo real que requieren otros procesadores, como los implicados en la recogida y etiquetado de datos.

No obstante, se echa en falta la posibilidad de seleccionar de forma dinámica (en tiempo de ejecución, sin tener que tocar el código fuente de la aplicación y compilarla de nuevo) algún tipo de procesado que aplicar a los datos.

4.1.4. Generación de clasificadores. Generación de archivos `arff`.

Para poder utilizar las funcionalidades del programa MindReader numeradas del 5 al 8 en la sección 4.1.1 (realizar una sesión de adquisición con retroalimentación o usar aplicación para el control de un cursor) se necesita un clasificador adaptado al usuario. En esta primera versión del sistema BCI la generación de clasificadores se hacía mediante el programa nnt, que permite entrenar redes de neuronas artificiales utilizando para el entrenamiento los ficheros de salida del programa MindReader que contienen los datos con la estimación del PSD. Los ficheros de salida generados por este programa se usaban posteriormente en el programa MindReader para realizar las sesiones de adquisición con retroalimentación o para utilizar la aplicación de control del cursor.

Para generar archivos arff para analizar con WEKA los datos recogidos en las sesiones de adquisición era necesario utilizar el script para Octave (ver apartado 3.5) o utilizar el programa nnt (ver apartado 3.3).

4.1.5. Parámetros utilizados

En esta sección se van a detallar los parámetros utilizados por la aplicación MindReader y por la aplicación nnt.

En la aplicación MindReader se deben introducir los parámetros que se describen a continuación, agrupados según la/s ventana/s donde aparecen.

Ventana de inicio:

Servidor Dirección IP donde se encuentra el servidor de datos que envía los datos a la aplicación. Si se usa el programa *Brain Vision Recorder* y éste está instalado en la misma máquina donde corre la aplicación, será la dirección de `localhost` (normalmente `127.0.0.1`).

Puerto Puerto en el que corre el servidor de datos. Si se usa el programa Brain Vision Recorder será el puerto `51244` para obtener los datos con precisión simple o el puerto `51234` para obtener los datos con precisión doble. El protocolo de transporte utilizado será TCP.

Parámetros comunes a todas las ventanas de configuración de sesiones de adquisición (sonoras o auditivas, con o sin retroalimentación)

Patrones a ignorar en cada clase Número de patrones a descartar del principio de cada clase. Debería ser igual o superior al número de patrones generados durante el tiempo de pausa. El valor de este parámetro quedará reflejado en el fichero de salida de sufijo `_patterns.mrp`, realmente el programa MindReader sólo lo usa al escribirlo en dicho fichero. El programa nnt lo utilizará al cargar los datos del fichero para utilizarlos en el entrenamiento de una red de neuronas o para generar un archivo `.arff` que pueda ser procesado con el programa WEKA.

Solapamiento El parámetro *solapamiento* indica cuántos instantes de tiempo (o muestras) no se utilizan entre el cálculo de un patrón en el dominio de la frecuencia y el siguiente. Para calcular el PSD del instante n se necesitan las muestras desde la *n-num_muestras_fft* hasta la n, teóricamente el siguiente patrón se correspondería con el instante *n+solapamiento*, que se calcularía con los datos de la muestra *(n+solapamiento)-num_muestras_fft* a la muestra *n+solapamiento*. En realidad, el programa MindReader utiliza siempre un solapamiento de 1 a la hora de estimar el PSD, lo que quiere decir que calcula un patrón por cada muestra recogida. El parámetro *solapamiento* sólo se utiliza en el programa MindReader para escribirlo en el fichero de salida de sufijo `_patterns.mrp`, y sólo es utilizado por el programa nnt a la hora de cargar los datos del fichero.

Aunque se calcule un patrón por cada muestra recogida, a la hora de entrenar un clasificador o generar un archivo `*.arff` con el programa nnt, no se toman todos los datos del fichero del dominio de la frecuencia (`_psd.raw`). Si se hiciera esto probablemente se produciría pérdida de generalización y sobreaprendizaje durante el entrenamiento debido a la similitud entre los patrones. Se usan sólo un patrón cada *solapamiento* líneas del fichero, el resto de patrones se descartan. Esto es equivalente a haber calculado el PSD con el valor del parámetro *solapamiento* que haya indicado en usuario.

Patrones por clase Número de patrones en el dominio de la frecuencia que se generan cada vez que se muestra una clase.

Como se comentará más detalladamente en la sección 5.1, estos parámetros están muy relacionados entre sí, y con los parámetros *duración de la clase* y *duración de la pausa*, y su valor debe ajustarse al valor del resto, aplicando las fórmulas 5.1, 5.2, 5.3 y 5.4. Dado que el programa MindReader no utiliza estos parámetros para realizar ningún cálculo (*patrones a ignorar en cada clase*, *patrones por clase* y *solapamiento*), si los valores de estos parámetros no son adecuados, ésto no afecta a la integridad de los ficheros de datos obtenidos.

Antes de utilizar el programa nnt para generar un archivo `*.arff`, o para entrenar un clasificador a partir de los datos en el dominio de la frecuencia que se encuentran almacenados en un fichero, se puede modificar a mano el fichero `*_patterns.mrp` para indicar un número diferente de *patrones por clase* y *patrones a descartar* o de *solapamiento* que el que se especificó inicialmente a través del programa MindReader al realizar la sesión de adquisición de datos.

Duración de la clase Segundos durante los que el usuario debe pensar en una determinada clase.

Duración de la pausa Segundos que dura la pausa que se realiza entre que se muestra una clase y otra. En las sesiones de adquisición con interfaz visual el usuario no percibe ningún cambio en la interfaz hasta que termina el tiempo de pausa y se muestra la siguiente clase, es decir, la clase se muestra durante *duración de la clase* segundos + *duración de la pausa* segundos, y luego se cambia a la siguiente. En el caso de las sesiones de adquisición con interfaz auditiva, al finalizar el tiempo de *duración de la clase* se indica al usuario mediante un estímulo auditivo que deje de realizar el proceso mental correspondiente y se prepare para escuchar la siguiente indicación. Al cabo de *duración de la pausa* segundos se emite el sonido indicativo de realizar el siguiente proceso mental.

Fichero para salvar la sesión Prefijo que se usará para nombrar los ficheros de salida del programa MindReader. El programa genera tres ficheros, cuyo nombre resulta de concatenar dicho prefijo con los sufijos *_patterns.mrp*, *_time.raw* y *_psd.raw*.

Nombre de la clase Nombre identificativo de determinada clase (proceso mental a realizar por el usuario). Cuando se indique al usuario que debe realizar determinado proceso mental, se mostrará este valor. Se

pueden utilizar hasta cinco procesos mentales en una sesión, sólo se deben rellenar tantos campos como clases se vayan a utilizar.

Número de muestras para la FFT Tamaño de la ventana que se utilizará en el método de Welch para la estimación del PSD.

Frecuencia máxima Frecuencia máxima a utilizar de entre las calculadas. (Las frecuencias superiores serán descartadas).

Frecuencia mínima Frecuencia mínima a utilizar de entre las calculadas. (Las frecuencias inferiores serán descartadas).

Parámetros exclusivos de las ventanas de adquisición con interfaz visual (con o sin retroalimentación)

RGB clase Color con el que se asocia determinada clase (proceso mental a realizar por el usuario). Son seis dígitos hexadecimales, los dos de la izquierda representan la cantidad de componente roja del color, los dos de en medio se corresponden con la componente verde y los dos de la derecha con la componente azul. Existen cinco campos de este tipo, cada uno se corresponde con un campo de tipo *nombre de la clase*. Sólo deben rellenarse tantos campos como clases se vayan a utilizar.

Parámetros exclusivos de las ventanas de adquisición con interfaz auditiva (con o sin retroalimentación))

Sonido Sonido con el que se asocia determinada clase (proceso mental a realizar por el usuario). Será la ruta del fichero que contiene dicho sonido. Existen cinco campos de este tipo, cada uno se corresponde con un campo de tipo *nombre de la clase*. Sólo deben rellenarse tantos campos como clases se vayan a utilizar.

Parámetros exclusivos de las ventanas de adquisición con retroalimentación y de las ventanas de clasificación (con interfaz auditiva o sonora)

Ruta del clasificador Ruta del fichero que contiene la especificación del clasificador a utilizar. El programa MindReader está preparado para utilizar clasificadores de tipo perceptrón multicapa de la librería FANN.

Clasificar cada Número de muestras que se obviarán entre una clasificación y otra. Para no sobrecargar la aplicación durante las sesiones de clasi-

ficación o de adquisición con retroalimentación, sólo se calcula y se muestra el resultado de clasificar un patrón cada X muestras.

Umbral Sólo se muestra al usuario el resultado del clasificador, cuando la clasificación supera cierto umbral de certeza. En el caso de utilizar redes neuronales con tantas neuronas de salida como clases (donde cada neurona de salida está asociada a una clase), la salida del clasificador es la clase que se corresponde con la neurona que ha obtenido un valor más alto. Si dicha neurona no supera el umbral especificado, no se muestra la salida al usuario (en el caso de las sesiones de adquisición con retroalimentación) o no se realizará acción alguna en las sesiones de clasificación. Se puede especificar un umbral diferente para cada clase.

Parámetros exclusivos de las ventanas de clasificación (con interfaz auditiva o sonora)

Acción La aplicación para comprobar el funcionamiento del clasificador es un cursor con forma de apuntador que se mueve por una ventana mediante dos acciones, avanzar (AC_FORDWARD) y rotar (AC_ROTATE). Además se permite realizar otra acción, (AC_CLICK) representada mediante un cambio del color del cursor. A cada clase que maneje el clasificador que se vaya a utilizar se asocia una acción del cursor. Cuando la salida de la neurona ganadora supere el umbral de la clase correspondiente, se realizará la acción asociada a a dicha clase.

Parámetros de la aplicación MindReader no configurables de forma dinámica

Hay ciertos parámetros que utiliza la aplicación que no pueden ser modificados en tiempo de ejecución, habría que modificar el código fuente y volver a compilar el programa. Son los siguientes:

Frecuencia de muestreo La frecuencia de muestreo, es decir, cuantas muestras recibe la aplicación cada segundo, es un parámetro que está fijado a 500 Hz en el código fuente de la aplicación.

Número de canales El número de canales por los que se recibe las señales también está fijado en el código fuente. Se utilizan 8 canales, ya que es el número de canales de los que consta el amplificador de señal V-Amp que se utiliza en el sistema BCI.

Longitud de los canales Número de frecuencias que se utilizan por cada canal una vez calculado el PSD. El valor adecuado para este parámetro depende del número de muestras para la FFT y el valor de la frecuencia máxima y mínima a utilizar. Sin embargo, está fijado en el código fuente a 12, valor que se corresponde con usar 250 muestras para el cálculo de la FFT y 8 Hz de frecuencia mínima y 30 Hz de frecuencia máxima.

Solapamiento Como ya se ha comentado anteriormente, aunque el usuario introduce un valor para el parámetro *solapamiento*, que quedará reflejado en el fichero de salida `*_patters.mrp` y que sí que será utilizado por la aplicación nnt, MindReader en realidad utiliza un solapamiento de 1 muestra a la hora de aplicar el método de Welch para el cálculo del PSD. Esto es útil para poder depurar la aplicación más fácilmente, aunque provoca que el procesado sea más lento de lo que podría ser.

Parámetros utilizados por la aplicación nnt

Los parámetros que se utilizan en la aplicación nnt son los que se encuentran en el fichero `*_patters.mrp` de donde va a cargar los datos y los que se especifican en el fichero de configuración que se le pasa como parámetro por la línea de comandos al invocar la aplicación. En la figura 4.4) se muestra un ejemplo de fichero de configuración, en la figura 4.5 se muestra un ejemplo del inicio de un fichero `*_patters.mrp` (faltan el resto de líneas que indican dónde comienzan los bloques de datos de cada clase en los ficheros `*.raw`).

```
1 RUN_ID david_s2_02
2 OUTPUT_PATH C:\Maite\Proyecto\Mindreader_1.0\tmp\sesion_2\
3 ARCH 13;5
4 LEARN_RATE 0.15
5 LEARN_MOMENTUM 0.85
6 TRAIN_PERC 20
7 FILE_PREFIX C:\Maite\Proyecto\Mindreader_1.0\tmp\sesion_2\
      s2_david_02
8 CYCLES 20000
9 ERROR_REPORT 10
```

Figura 4.4: Ejemplo de archivo de configuración válido para el programa nnt

Los parámetros contenidos en el fichero `*_patters.mrp` provienen del programa MindReader, y ya se ha explicado su uso anteriormente. Es importante que la palabra clave que identifica a cada comando esté escrita según se especifica en el fichero `ConfigConst.h`, respetando las mayúsculas. La co-

```
1  NUMBER\_OF\_CHANNELS 8
2  NUMBER\_OF\_CLASSES 2
3  SAMPLES\_TO\_SKIP 500
4  SAMPLES\_OVERLAP 31
5  CHANNEL\_LENGTH 12
6  SAMPLES\_PER_CLASS 300)
7
8  #mover mano
9  0 0
10 #rotar
11 10500 1
```

Figura 4.5: Ejemplo de archivo patterns.mrp (primeras líneas)

rrespondencia entre las palabras claves de este fichero y los parámetros que se introducen en el programa MindReader es la siguiente:

Número de canales Palabra clave: NUMBER_OF_CHANNELS.

Número de clases Palabra clave: NUMBER_OF_CLASSES.

Este parámetro lo genera automáticamente el programa MindReader según el número de clases que se haya especificado que se van a utilizar, es decir el número de campos *nombre de clase* que se haya rellenado.

Patrones a ignorar Palabra clave: SAMPLES_TO_SKIP.

Solapamiento Palabra clave: SAMPLES_OVERLAP.

Longitud del canal Palabra clave: CHANNEL_LENGTH.

Patrones por clase Palabra clave: SAMPLES_PER_CLASS.

Los parámetros que se especifican en el fichero de configuración cuya ruta se pasa en la línea de comandos al invocar el programa nnt se comentan a continuación. Hay que respetar la ortografía y mayúsculas del identificador del parámetro para que los datos de configuración puedan cargarse desde el fichero.

RUN_ID Los ficheros creados durante la ejecución tendrán como prefijo el identificador de ejecución. Los sufijos de los fichero de salida son `.arff` en el caso de que se haya invocado al programa con la opción `nnt arff`, o si se ha invocado con la opción `nnt net`, el fichero que contiene la red de neuronas generadas tomará el sufijo `.net` y el fichero con el informe de errores `_errors.txt`.

OUTPUT_PATH Directorio en el que se almacenarán el/los fichero/s de salida.

ARCH Arquitectura de la red de neuronas artificiales a utilizar, especificando únicamente el número de neuronas de las capas internas, separado por ";". Este parámetro sólo se tiene en cuenta si se invoca al programa con la opción `nnt net`.

LEARN_RATE Tasa de aprendizaje a utilizar durante el entrenamiento de la red de neuronas, para el algoritmo de retropropagación. Este parámetro sólo se tiene en cuenta si se invoca al programa con la opción `nnt net`.

LEARN_MOMENTUM Momento de aprendizaje de aprendizaje a utilizar durante el entrenamiento de la red de neuronas, para el algoritmo de retropropagación. Este parámetro sólo se tiene en cuenta si se invoca al programa con la opción `nnt net`.

TRAIN_PERC Porcentaje del conjunto de datos que se utilizará para entrenamiento. El resto de los datos será utilizado para componer el conjunto de validación. Este parámetro sólo se tiene en cuenta si se invoca al programa con la opción `nnt net`.

FILE_PREFIX Prefijo de los ficheros de donde se van a cargar los datos, separados por ";". La ruta debe ser la ruta completa a los ficheros.

CYCLES Número de ciclos de entrenamiento que se van a realizar. Este parámetro sólo se tiene en cuenta si se invoca al programa con la opción `nnt net`.

ERROR_REPORT Cada cuántos ciclos de aprendizaje se genera el informe de errores que se almacena en el archivo `_errors.txt`. Este parámetro sólo se tiene en cuenta si se invoca al programa con la opción `nnt net`.

4.1.6. Sobre la aleatorización de los datos en loadPatterns

Los datos contenidos en los ficheros `*_time.raw` y `*_psd.raw` se encuentran ordenados según fueron recogidos durante la sesión de adquisición a la que corresponden. Estos ficheros constan de una línea por cada instante de tiempo procesado. En un segundo se generan 500 líneas (tantas como el

número de muestras que se toman por segundo, es decir, la frecuencia de muestreo de la señal recogida con el casco encefalógrafo).

Al procesar con Octave los datos para generar los archivos `*.arff` que emplea WEKA, éstos se mantienen en el orden en que aparecen en el fichero, descartándose los patrones que se indiquen en el archivo `*_patterns.mrp`. Esto es, los primeros patrones de cada clase (PATTERNS_TO_SKIP en el fichero) y tomando un patrón cada *solapamiento* (SAMPLES_TO_OVERLAP en el fichero) patrones.

Al procesar los datos con el programa `nnt`, tanto para generar los ficheros `*.arff` como para generar los conjuntos de entrenamiento y validación que se emplean para entrenar las redes de neuronas, además de normalizarse los datos, se desordenan. Todo esto se realiza desde la función global `loadData`, que invoca sucesivas veces a los métodos `DataSet::loadPatterns` y `DataSet::initData`. Este desorden no es aleatorio, de hecho, en realidad mayoritariamente los datos siguen ordenados, simplemente están salteados. En esta sección se va a explicar el proceso de generación de dichos conjuntos para que se entienda la problemática.

El método `DataSet::loadPatterns` se limita a cargar los datos de un fichero determinado, descartando los patrones pertinentes según los parámetros especificados en el mismo.

El algoritmo básico para generar los conjuntos de entrenamiento y de validación, implementado en el método `DataSet::initData` es el siguiente:

1. El porcentaje de entrenamiento y de test dependen del valor que se haya indicado. Se dispone de un conjunto con todos los patrones de un fichero, habiendo ya eliminado los patrones a descartar del principio de cada clase, y de los conjuntos de entrenamiento y de test, inicialmente vacíos.

2. Se calculan cuántos elementos del conjunto total de patrones deben ir al conjunto de entrenamiento y cuántos al de validación según el porcentaje de test que se ha indicado.

3. Se simula la tirada de un dado de dos caras, se saca el primer elemento del conjunto total de patrones, y según el resultado del dado, se inserta en el conjunto de entrenamiento o de test.

4. Se realiza el paso 3 hasta que los dos conjuntos tienen el número adecuado de patrones.

Esto implica que uno de los dos conjuntos se rellenará antes (previsiblemente el de test, que contiene menos elementos) y a partir de que dicho

conjunto esté completo, el resto de los patrones será insertado en orden, uno detrás de otro, en el otro conjunto.

Los datos realmente sí que están algo desordenados, ya que este algoritmo no se aplica una única vez sino varias veces, volcando después de cada aplicación (cada vez que se carga un fichero) los conjuntos de entrenamiento y test al conjunto global (además, durante dicho volcado, el orden de los patrones se invierte). Aún así, los datos no han sido aleatorizados, aunque algunos se encuentren desordenados. Los patrones se encuentran más o menos ordenados por orden de aparición (ya se inverso o no) y por ficheros.

El algoritmo final que se emplea cuando se invoca al programa nnt indicándole que tome los patrones de uno o varios ficheros, implementado en la función `loadData`, es el siguiente:

- Se dispone de un conjunto (`DataSet`) final y otro auxiliar. Por cada fichero:
  - Se cargan los datos del fichero en el conjunto auxiliar.
  - Se generan el conjunto de entrenamiento y validación según el algoritmo anterior, tomando siempre como porcentaje de test un 30 %.
  - Se insertan en el conjunto final, introduciendo primero el conjunto de entrenamiento y luego el de test. En este proceso el orden de los patrones queda invertido.
- Cuando se ha terminado de realizar esto con todos los ficheros indicados, se aplica de nuevo el algoritmo, esta vez sobre el conjunto que contiene los patrones de todos los ficheros y teniendo en cuenta el porcentaje de patrones para el conjunto de entrenamiento especificado por el usuario.
- Por último, en el caso de estar generando el fichero arff, se escribe la cabecera del fichero, a continuación los datos de entrenamiento y por último los del conjunto de test. En el caso de estar entrenando una red, simplemente se le pasan los conjuntos de entrenamiento y de validación tal cual.

Por tanto, en los conjuntos de test probablemente aparezcan bastantes datos de los primeros ficheros, pero ninguno de los del final.

Al no aleatorizarse los datos, y al no poder asegurar que ambos conjuntos contienen datos de cada uno de los ficheros, puede ocurrir que en el conjunto de entrenamiento haya patrones muy similares entre sí y a su vez muy dispares

de los que se encuentren en el conjunto de validación. Esto podría provocar que el clasificador no consiguiese clasificar correctamente los patrones del conjunto de validación.

4.2. Aspectos a mejorar de la herramienta

La herramienta MindReader en su primera versión era una aplicación muy completa, pero se echaban de menos ciertos aspectos y funcionalidades que se enumeran a continuación:

1. Integración de las funcionalidades del programa nnt en el programa MindReader. Es decir, poder generar, entrenar y almacenar en un archivo redes de neuronas a partir de los datos del PSD contenidos en un fichero y a su vez poder generar archivos `arff` a partir de dicho fichero.

2. Posibilidad de reentrenar redes de neuronas de la librería FANN. Es decir, poder generar una nueva red de neuronas tomando la arquitectura de una red ya existente y usar como pesos iniciales de la nueva red los de la red antigua.

3. Posibilidad de que al finalizar la sesión de adquisición la generación de redes de neuronas se realice automáticamente entrenando la red a generar con los datos obtenidos durante la sesión de adquisición de datos recién realizada.

4. Una vez realizada la sesión de adquisición no se pueden alterar los parámetros relativos al cálculo de la estimación del PSD. Sería interesante que la aplicación permita simular sesiones, es decir, volver a generar el fichero de datos con la estimación del PSD (o realizar otra transformación que se desee aplicar a los datos) a partir de los datos en el dominio del tiempo de una sesión de adquisición, variando los parámetros para calcular el mismo. Esta nueva funcionalidad permitiría modificar los parámetros de frecuencia máxima y mínima y el número de muestras usadas en el cálculo de la FFT.

5. Posibilidad de aplicar un filtro a los datos en el dominio del tiempo. Los nuevos datos del dominio del tiempo se obtendrían de multiplicar los datos de cada línea del fichero por una matriz cuadrada de igual dimensión que el número de canales.

6. Permitir valores superiores a 256 en el tamaño de ventana que se utiliza en el método de Welch.

7. Implementación de nuevos procesadores y clasificadores. El programa MindReader está pensado para que sea flexible y fácilmente ampliable, por lo que se podrían implementar y utilizar otros procesadores u otros clasificadores. .

8. La aplicación nnt no permite entrenar un clasificador utilizando los datos de determinado fichero para entrenar y los datos de otro para validar. A la hora de entrenar un clasificador sólo se permite indicar qué fichero/s se van a utilizar y qué porcentaje de los mismos se quiere utilizar como conjunto de validación. A partir de estos datos carga los ficheros y calcula el conjunto de validación y el de entrenamiento de forma que disponga cada uno del porcentaje adecuado de datos, eligiendo qué dato se ubica en cada conjunto de una forma pseudoaleatoria.

9. No se dispone de un instalador para la aplicación.

10. Los primeros valores que se obtienen en un periodo de adquisición no son fiables porque el buffer de la aplicación se encuentra vacío y el PSD calculado no es válido hasta que se disponen de tantos datos como el tamaño de la ventana usada al calcular el PSD. Sería interesante que la aplicación empezara a calcular el PSD una vez que el buffer tiene suficientes datos.

11. Se producen fallos de memoria (a veces la aplicación finaliza de forma inesperada) que deberían ser controlados. Además se han detectado pérdidas de memoria.

12. Al generar la aplicación desde el IDE *Microsoft Visual Studio 2005*, la versión generada en modo "release" no arranca, sólo puede utilizarse la aplicación generada en modo "debug".

13. Algunos aspectos relativos a la usabilidad de la aplicación deberían ser mejorados, como son el control de completitud y formato de los parámetros introducidos, la falta de uniformidad en los títulos de las diferentes ventanas y la carencia de cuadros de progreso durante operaciones de larga duración.

4.3. Mejoras realizadas sobre la aplicación

En el apartado 4.2 se enumeraron una serie de aspectos que se podrían mejorar de la herramienta MindReader. Al añadir las funcionalidades a las

que se accede desde la nueva ventana de «Entrenamiento de redes y simulación de sesiones» se cubrieron los puntos 1, 2, 3, 4, 5, 7 de dicha lista, teniendo especial cuidado en no cometer errores que fueran en detrimento de la usabilidad del sistema. En la sección 4.4 se explica detalladamente este añadido a la aplicación, que es la aportación principal a la antigua versión del programa MindReader realizada durante el desarrollo de este proyecto de fin de carrera.

Además se han realizado otras mejoras en la aplicación, que se enumeran a continuación.

- Al compilar el código fuente de la aplicación original se producían 30 *warnings*. Se ha corregido el código fuente para que ésto no ocurra.

- Se ha eliminado la restricción que limitaba el tamaño de ventana del método de Welch para el cálculo del PSD a 256 elementos (Elemento 6 de la lista de aspectos a mejorar).

- Se ha generado un instalador que permite instalar de forma cómoda la aplicación en un entorno MS Windows (Elemento 9 de la lista de aspectos a mejorar).

- Se han eliminado algunas pérdidas de memoria liberando memoria mediante la llamada `free` en algunos puntos del código fuente, no obstante siguen produciéndose pérdidas de memoria cuya causa no ha conseguido detectarse.(Elemento 11 de la lista de aspectos a mejorar).

- Como se comentó anteriormente en la sección 4.1.6, en la primera versión del sistema BCI al utilizar el programa nnt, al cargar cada fichero se desordenaban los datos un 30 %, independientemente del porcentaje de datos que hubiese especificado el usuario que debía utilizarse para constituir el conjunto de entrenamiento, según el algoritmo explicado en dicho apartado. Esto ya no ocurre con el programa MindReader, se toma el porcentaje especificado por el usuario, lo que permite que al generar archivos `arff` para procesarlos con WEKA, los datos puedan estar ordenados de la misma forma que aparecen en los ficheros `_psd.raw` de origen.

Se ha intentado solucionar otros aspectos, pero no se ha conseguido, como es el generar una versión funcional del programa en modo *release* en lugar de en modo *debug* desde el IDE *Microsoft Visual Studio 2005*. Probablemente es debido a las pérdidas de memoria que se producen y que no se han podido detectar.

4.4. Nuevas funcionalidades. Entrenamiento redes-Simulación sesiones.

4.4.1. Funcionalidades y uso

El acceso a las nuevas funcionalidades añadidas a la aplicación relativas al entrenamiento de redes, simulación de sesiones y aplicación de filtros espaciales, está centralizado en una única ventana, que denominamos ventana de "Entrenamiento de redes".

Se añadió un nuevo botón con el rótulo "Entrenamiento de Redes" a la ventana de inicio de la aplicación, en la figura 4.6 se muestra dicha ventana tal y como era en la aplicación inicial y en su actual forma.

Pulsando el nuevo botón aparece la ventana de "Entrenamiento de redes". En la figura 4.7 se muestra el aspecto de dicha ventana y se especifican los controles que puede modificar el usuario.

Existen varios grupos de controles, cada grupo contiene los parámetros relativos a una funcionalidad distinta. Las nuevas funcionalidades del sistema a las que se puede acceder a partir de esta interfaz son las siguientes:

- Realizar una nueva sesión de adquisición, especificando los parámetros de la misma en los controles 23 a 45 (ver figura 4.7). Para seleccionar esta acción y que se habiliten las casillas de los parámetros correspondientes, hay que marcar el botón de tipo radio que aparece en la figura 4.7 con el número 10, *Realizar sesión de adquisición de datos*. Se puede especificar un filtro que aplicar a los datos en el dominio del tiempo, antes de calcular el PSD, marcando el botón 26 y especificando en la casilla 28 la ruta del fichero que contiene el filtro. El filtro consistirá en una matriz cuadrada de dimensión igual al número de canales empleados.

- Simular una sesión de adquisición a partir de los datos de la señal en el dominio del tiempo contenidos en uno o varios ficheros obtenidos en sesiones anteriormente realizadas. Gracias a esta funcionalidad se pueden variar los parámetros relativos al cálculo del PSD y filtrado de frecuencia especificados cuando se realizó la sesión de adquisición, y generar de nuevo el fichero con los datos en el dominio de la frecuencia, pero empleando en su generación los nuevos parámetros. En este caso hay que marcar el botón de tipo radio 9, *Simular sesión de entrenamiento a partir de los datos del archivo/s*, para que se habiliten los campos que se deben rellenar (22 a 28). Hay un detalle importante a tener en cuenta si se selecciona esta opción, los parámetros CLASS_TIME

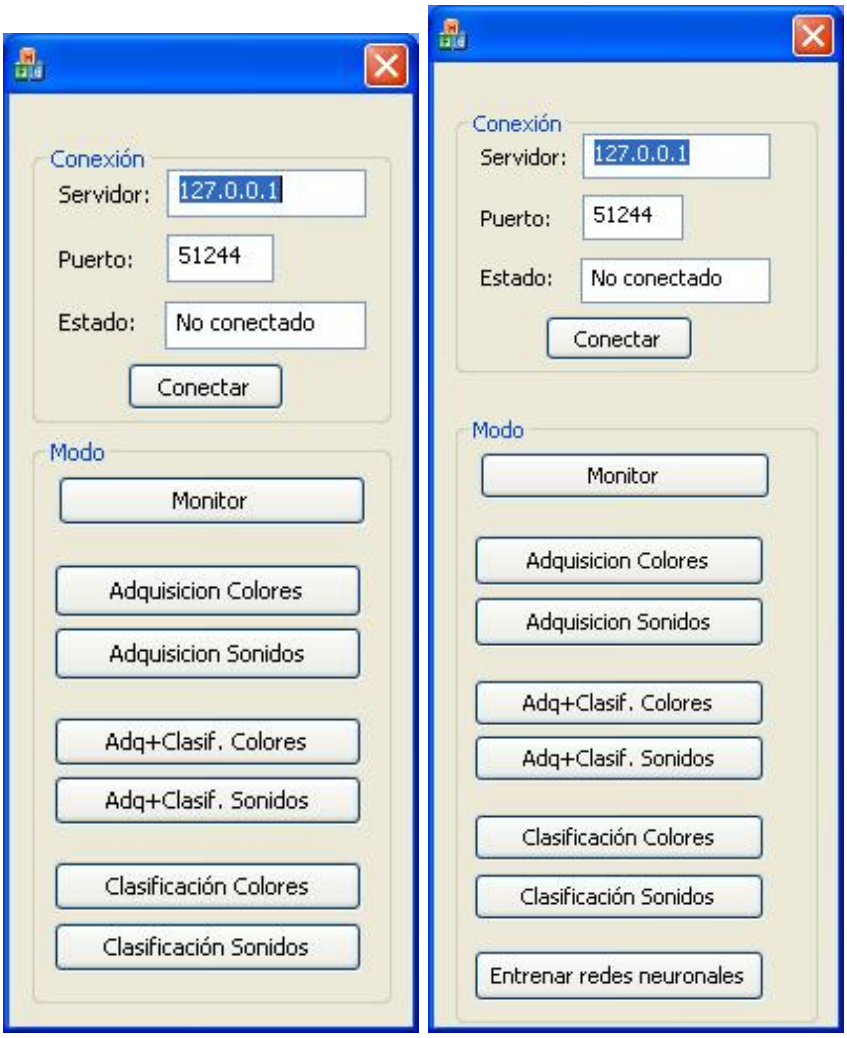

Figura 4.6: Pantalla inicial MindReader. Versión antigua (izqda.) y versión nueva (dcha.)

y PAUSE_TIME deben constar en el/los fichero/s *_patterns.mrp* cuyo prefijo se ha especificado en la casilla 22. A la hora de simular una sesión de adquisición se puede también especificar un filtro que aplicar a los datos en el dominio del tiempo antes de calcular el PSD (utilizando los controles 26 a 28).

- Generar los archivos `.arff` con los datos en el dominio de la frecuencia de los ficheros cuyo prefijo se especifique en la casilla 22, para que puedan ser procesados en WEKA. En este caso hay que marcar el

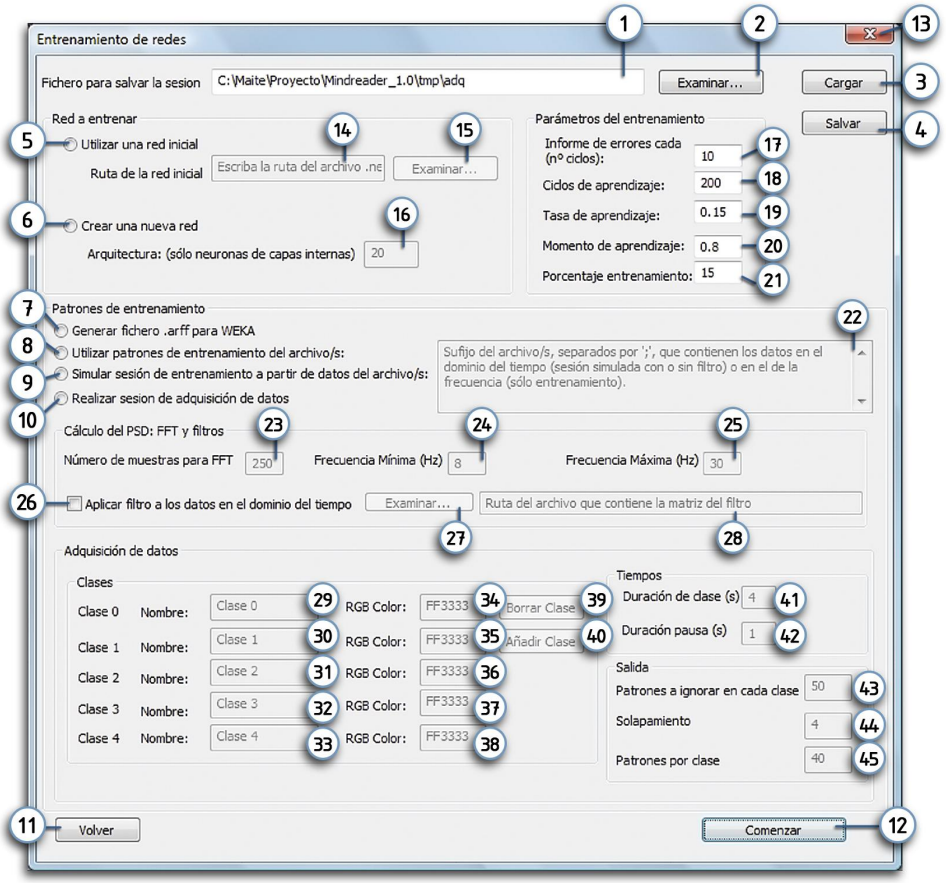

Figura 4.7: Interfaz entrenamiento redes y simulación de sesiones

botón de tipo radio 7, *Generar fichero arff para WEKA*. El resultado es equivalente a haber invocado al programa nnt con la opción `nnt arff`.

- Utilizar los ficheros con los datos en el dominio de la frecuencia obtenidos en una sesión de adquisición realizada anteriormente (cuyo nombre se especificará en la casilla 22) para entrenar una red de neuronas. En este caso hay que marcar el botón de tipo radio 8, *Utilizar patrones de entrenamiento del archivo*. Si lo que se genera es una nueva red de neuronas cuya arquitectura se especifica por parámetros, esto es equivalente a haber invocado el programa nnt con la opción `nnt net`.

Tanto si se procede con la realización de una nueva sesión de adquisición de datos o con una simulación o directamente con la carga de ficheros que

```
FANN_FLO_2.1
num_layers=3
learning_rate=0.150000
connection_rate=1.000000
network_type=0
learning_momentum=0.800000
training_algorithm=2
train_error_function=1
train_stop_function=0
cascade_output_change_fraction=0.010000
quickprop_decay=-0.000100
quickprop_mu=1.750000
rprop_increase_factor=1.200000
rprop_decrease_factor=0.500000
rprop_delta_min=0.000000
rprop_delta_max=50.000000
rprop_delta_zero=0.100000
cascade_output_stagnation_epochs=12
cascade_candidate_change_fraction=0.010000
cascade_candidate_stagnation_epochs=12
cascade_max_out_epochs=150
cascade_max_cand_epochs=150
cascade_num_candidate_groups=2
bit_fail_limit=3.49999999999999980000e-001
cascade_candidate_limit=1.0000000000000000000e+003
cascade_weight_multiplier=4.0000000000000020000e-001
cascade_activation_functions_count=10
cascade_activation_functions=3 5 7 8 10 11 14 15 16 17
cascade_activation_steepnesses_count=4
cascade_activation_steepnesses=2.5000000000000000000e-001
    5.0000000000000000000e-001 7.5000000000000000000e-001
    1.0000000000000000000e+000
layer_sizes=97 21 3
scale_included=0
neurons (num_inputs, activation_function, activation_steepness)=(0,
     0, 0.00000000000000000000e+000) (0, 0, 0.00000000000000000000e
    +000) (0, 0, 0.00000000000000000000e+000) (0, 0,
    0.00000000000000000000e+000) ...(0, 3, 0.00000000000000000000e
    +000) (21, 3, 5.00000000000000000000e-001) (21, 3,
    5.00000000000000000000e-001) (0, 3, 0.00000000000000000000e
    +000)
connections (connected_to_neuron, weight)=(0,
    -1.58525659035439240000e+001) (1, -1.62823972642158380000e+001)
     (2, -1.89108013315744970000e+001) (3, -7.95345852931664240000e
    +002) (4, -9.85669798489318400000e+002) (5,
    -7.51325029205229670000e+001) ... (114, 1.50000000000000000000e
    +003) (115, 2.80286247192165160000e-001) (116,
    1.24401999934137850000e+000) (117, 9.50431674843528610000e-002)
```

Figura 4.8: Ejemplo de archivo .net con la especificación de una red de neuronas de la librería FANN

contengan los datos con el PSD, los datos del PSD obtenidos pueden utilizarse para entrenar una red de neuronas, que podrá ser:

- Una nueva red de neuronas, especificando su arquitectura y los parámetros que se van a utilizar para su entrenamiento. Los pesos iniciales asignados a cada par de neuronas antes de comenzar el entrenamiento se ajustarán automáticamente de forma aleatoria. La arquitectura de la nueva red debe especificarse en la casilla 16, indicando sólo el número de neuronas de cada capa interna separado por ":", al igual que se hacía en el programa nnt. Para realizar esta acción deberá marcarse el botón de tipo radio 6 *Crear una nueva red*.

- Una nueva red de neuronas a partir de una ya existente, especificando los parámetros que se van a utilizar para su entrenamiento. Los pesos iniciales y la arquitectura de la nueva red serán los de la red de partida, cuya especificación se encuentra en un fichero cuyo nombre se indicará en la casilla 14. Para realizar esta acción deberá marcarse el botón de tipo radio 5 *Utilizar red inicial*.

En ambos casos, los parámetros del entrenamiento a realizar deben especificarse en las casillas 17 a 21 (ver figura 4.7).

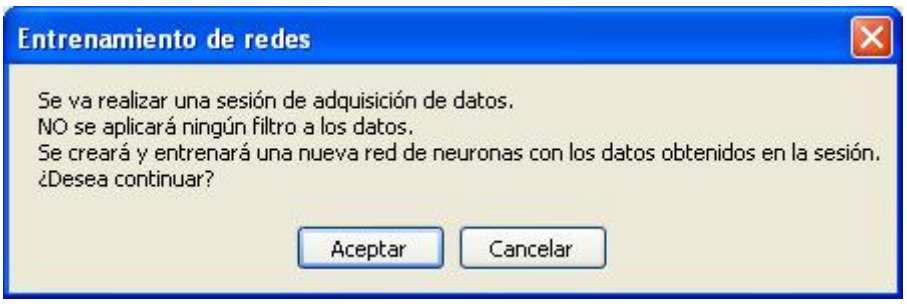

Figura 4.9: Cuadro de diálogo de confirmación de la acción a realizar (I)

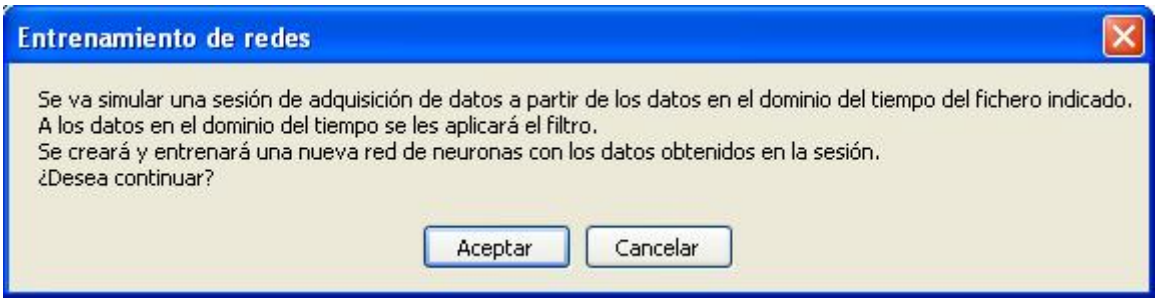

Figura 4.10: Cuadro de diálogo de confirmación de la acción a realizar (II)

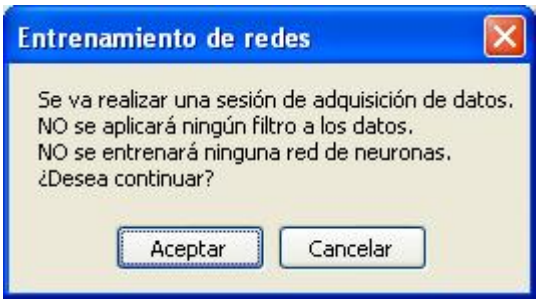

Figura 4.11: Cuadro de diálogo de confirmación de la acción a realizar (III)

Para tomar una red ya existente en lugar de especificar los parámetros de la red nueva, en este caso se debe especificar el fichero donde se encuentra almacenada dicha red. El formato del fichero debe ser el mismo que utiliza la librería FANN cuando almacena una red de neuronas, lo ideal es utilizar una red que haya sido generada y almacenada en un fichero mediante dicha librería, aunque también podría generarse el fichero manualmente o desde una aplicación externa. El formato de estos ficheros es el que se muestra en la figura 4.8. Nótese que se han omitido valores de las líneas 33 y 34 del fichero, donde se especifican las neuronas de la red artificial y los pesos de las conexiones entre ellas ya que eran demasiado extensas.

Una vez seleccionadas las acciones a realizar (mediante los botones de tipo radio 5 a 10) y una vez especificados los parámetros en las casillas correspondientes, se pulsa el botón 12, *comenzar*. Un cuadro de diálogo pedirá confirmación al usuario, y si este no lo cancela, la acción se llevará a cabo. En las figuras 4.9, 4.10 y 4.11 se muestran varios ejemplos de los cuadros de diálogo de confirmación. Si se ha seleccionado la opción de *Realizar una nueva sesión de adquisición de datos* (botón 10), la ventana correspondiente (ver figura 4.12) aparecerá, si no, se mostrará un cuadro de progreso indicando el avance del proceso que se está realizando (ver figuras 4.13 y 4.14).

Mediante el botón 4, *Salvar*, se puede guardar la configuración especificada por pantalla en un fichero, para poder reutilizarla más adelante. Para cargar los valores de los parámetros guardados de esta forma, debe seleccionarse un fichero de configuración tras pulsar el botón 3, *Cargar*. El formato de estos ficheros es el que se muestra en la figura 4.15. Cada línea del fichero es un par de palabras, siendo la primera la clave que identifica al atributo y la segunda el valor asignado a dicho atributo. Las claves para identificar los diferentes atributos se encuentran en el fichero cabecera `Const.h`, si se modifica manualmente el fichero de configuración debe tenerse en cuenta que las claves deben escribirse exactamente igual como aparecen en dicho fichero, en caso contrario la aplicación no será capaz de identificar de qué parámetro

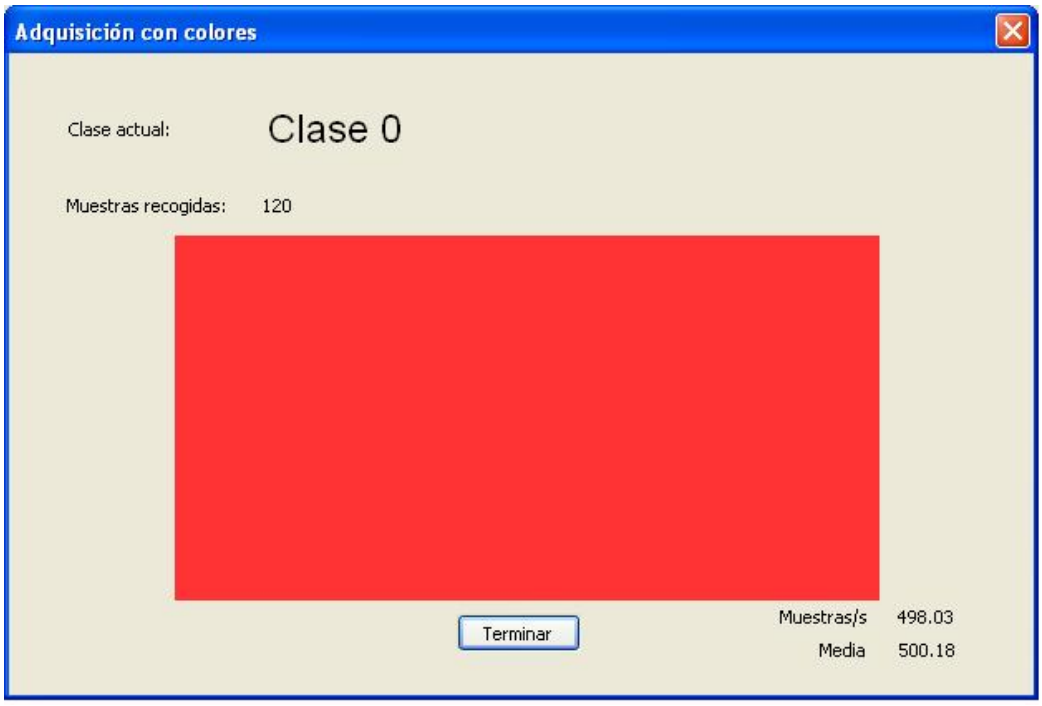

Figura 4.12: Ventana de sesión de adquisición con interfaz visual

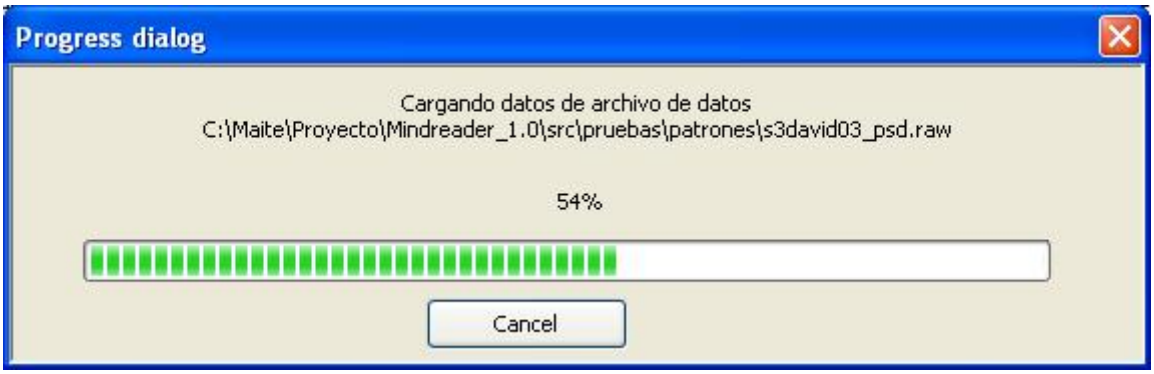

Figura 4.13: Cuadro de diálogos de progreso (I)

se trata y se producirán errores en tiempo de ejecución.

El botón 11, *volver*, hace que se cierre la ventana de *Entrenamiento de Redes* (figura 4.7) y vuelva a aparecer la ventana inicial de la aplicación (figura 4.6).

Figura 4.14: Cuadro de diálogo de progreso (II)

```
1
2  ARCH 20
3  CLASSES_COLORS FF5533;3355FF;FF3333;FF3333
4  CLASSES_NAMES Clase|0;Clase|1;Clase|2;Clase|3
5  CLASSIFIER_PATH C:\Maite\Proyecto\Mindreader_1.0\src\pruebas\
       david_s3_p3.net
6  CLASS_LENGTH 15
7  CYCLES 200
8  ERROR_REPORT 10
9  FILE_PREFIX C:\Maite\Proyecto\Mindreader_1.0\src\pruebas\patrones\
       s3david03;C:\Maite\Proyecto\Mindreader_1.0\src\pruebas\patrones
       \s3david04
10 FILTER_FILE C:\Maite\Proyecto\Mindreader_1.0\src\pruebas\
       prueba_filtro.txt
11 LEARN_MOMENTUM 0.8
12 LEARN_RATE 0.15
13 MAX_FREQ 30
14 MIN_FREQ 8
15 NFFT 250
16 NUMBER_OF_CLASSES 4
17 OUTPUT_PATH Escriba|la|ruta|del|archivo|.net|que|contiene|la|red
18 PAUSE_LENGTH 1
19 RUN_ID C:\Maite\Proyecto\Mindreader_1.0\src\pruebas\
       prueba_RedInicial_simulate
20 SAMPLES_OVERLAP 31
21 SAMPLES_PER_CLASS 234
22 SAMPLES_TO_SKIP 500
23 TRAIN_PERC 85
```

Figura 4.15: Ejemplo de archivo de configuración para la carga de parámetros desde la interfaz de *Entrenamiento de Redes* de la aplicación MindReader

4.4.2. Ficheros de salida generados

Dependiendo de qué acción se seleccione para ser llevada a cabo, la aplicación MindReader generará unos ficheros de salida u otros. El inicio del

nombre de todos los ficheros de salida generados al utilizar las funcionalidades que se acceden desde esta ventana coincidirá con el especificado en la casilla 1, *Fichero para salvar la sesión* (ver figura 4.7). La ubicación de todos los archivos de salida será la misma, ya que el prefijo que contiene dicha casilla debe ser la ruta completa del directorio donde se almacenarán los ficheros de salida y el prefijo de dichos ficheros.

La versión inicial de la aplicación MindReader generaba tres ficheros al realizar una sesión de adquisición, cuyos sufijos eran `_patterns.mrp`, `_time.raw` y `_psd.raw`. En la nueva versión del programa se generan ficheros análogos a éstos que siguen su mismo formato, aunque el sufijo con el que se genera el nombre del archivo puede variar según se haya generado al ejecutar una funcionalidad u otra.

El fichero `*_time.raw` contiene los datos en el dominio del tiempo. Cada línea del fichero se corresponde con un instante de tiempo, cada columna -excepto la primera- representa un canal. Por lo tanto, cada línea consta de nueve columnas: en la primera figura el número de línea del fichero, y en las siguientes, para cada uno de los ocho canales utilizados, el valor de la señal transmitida por dicho canal en ese preciso instante de tiempo.

El archivo `*_psd.raw` contiene los datos con la estimación del PSD realizada utilizando los parámetros especificados por el usuario. Cada línea del fichero se corresponde con una línea del fichero en el dominio del tiempo `*_time.raw`, de forma que en este fichero se encuentra la estimación del PSD de los datos de cada uno de los instantes recogidos durante la sesión de adquisición.

El fichero `*_patterns.mrp` indica a qué clase corresponde cada línea de los ficheros `*_time.raw` y `*_psd.raw`, en los cuales bloques de datos consecutivos pertenecen a la misma clase. En este fichero se indica en qué línea comienza un bloque de datos perteneciente a determinada clase, además de los parámetros empleados para el cálculo del PSD.

La nueva versión del programa genera además ficheros con sufijos `.net` y `_NetErrors.txt` al realizar un entrenamiento de redes de neuronas. El primero contiene la especificación de la red generada por el programa según el formato de almacenamiento de redes de neuronas de la librería FANN (ver figura 4.8). El segundo contiene el informe de errores cada cierto número de ciclo de entrenamientos, mostrando por columnas el número de ciclo al que se corresponde el informe, el error al evaluar el conjunto de entrenamiento en dicho ciclo, el error al evaluar el conjunto de validación en dicho ciclo, el porcentaje de aciertos medio cometido por la red al evaluar el conjunto de validación, y por cada clase, el porcentaje de fallos cometidos (falsos positivos) al clasificar dicha clase. Se muestra un ejemplo de archivo generado durante un entrenamiento de 100 ciclos de duración clasificando entre dos

clases en la figura 4.16).

```
0 0   0.252831  0.501703  0   1
10  0.247248  0.251447  0.498297  1   0
20  0.245429  0.24995   0.498297  1   0
30  0.229799  0.244483  0.543947  0.371209  0.541477
40  0.227351  0.246932  0.527368  0.693526  0.250228
50  0.218644  0.251621  0.528958  0.618379  0.322698
60  0.214393  0.253935  0.527141  0.619285  0.325433
70  0.211669  0.255065  0.526686  0.609325  0.336372
80  0.208662  0.255198  0.531683  0.598461  0.337284
90  0.205341  0.254912  0.535317  0.611589  0.316773
```

Figura 4.16: Ejemplo de archivo con el informe de errores del entrenamiento de una red de neuronas de la librería FANN

```
#This is data from a simulated session. Old file:
# <nombre_fichero_origen>
```

Figura 4.17: Detalle del nuevo archivo patterns.mrp generado a partir de una sesión simulada

A continuación se especifica el nombre de los ficheros de salida que se generan según las acciones selecionadas:

- Ficheros de salida generados al realizar una sesión de adquisición.

  Al realizar una sesión de adquisición se genera un archivo de nombre `Prefijo_time.raw` que contiene los datos en el dominio del tiempo tal y como son recibidos del servidor de datos, sin aplicarles transformación alguna. También se genera el archivo `Prefijo_patterns.mrp` que se corresponde con el anterior.

  - Sin aplicar ningún filtro

    Si no se aplica ningún filtro, se generan además los archivos `Prefijo_psd.raw`, `Prefijo.net` y `Prefijo_NetErrors.txt`.
  - Aplicando filtros

    Se genera además el archivo `PrefijoFiltered_time.raw`, cuyas líneas se corresponde con aplicar el filtro a cada una de las líneas del archivo `Prefijo_time.raw`. Al calcular el PSD de

este archivo se generan los archivos `PrefijoFiltered_psd.raw` y `PrefijoFiltered_patterns.mrp`.

- Ficheros de salida generados al realizar una simulación de sesión de adquisición.

  - Sin aplicar ningún filtro

    Si no se aplica ningún filtro el nombre del fichero que contiene los datos del PSD es `PrefijoSimulated_psd.raw`. Cada línea del fichero se corresponde con una línea de los ficheros de origen, de forma que en este fichero se encuentra la estimación del PSD de los datos de todos los ficheros facilitados. Entre un bloque de datos generado a partir de los datos de un determinado fichero, y el siguiente bloque de datos, generado a partir de los datos del siguiente fichero, existen unas líneas comentadas a modo de separación, que se muestran en la figura 4.17. Se genera el archivo `PrefijoSimulated_patterns.mrp` que se corresponde con el anterior, y los archivos generados al entrenar la red de neuronas indicada `PrefijoSimulated.net` y `PrefijoSimulated_NetErrors.txt`.

  - Aplicando filtros

    El archivo `PrefijoFiltered_time.raw` contiene el resultado de aplicar el filtro a los datos de los ficheros de origen especificados. Entre un bloque de datos perteneciente a un fichero y otro, aparecen unas líneas comentadas que indican la procedencia del siguiente bloque de datos (ver figura 4.17). Se genera además el archivo `PrefijoFiltered_psd.raw` con el cálculo del PSD del archivo anterior, el archivo `PrefijoFiltered_patterns.mrp` con los metadatos que se corresponden con ambos archivos y los archivos `PrefijoFiltered.net` y `PrefijoFiltered_NetErrors.txt` con los resultados del entrenamiento de la red neuronal.

- Al entrenar una red de neuronas con los datos del PSD contenidos en un fichero.

  Se generan los archivos `Prefijo.net` y `Prefijo_NetErrors.txt`.

- Fichero de salida para WEKA

  Se genera el archivo `prefijo.arff` que contiene los datos de los archivos que se haya especificado en el formato que utiliza el programa WEKA.

4.4.3. Usabilidad de la interfaz

A la hora de diseñar e implementar la ventana a través de la cual se accede a las funcionalidades de simulación de sesiones de adquisición de datos y entrenamiento de redes, se ha tenido especial cuidado en mejorar la usabilidad de la interfaz con respecto al resto de ventanas del programa. De esta forma se espera que el usuario tenga una experiencia más agradable a la hora de utilizar el programa MindReader.

A continuación se enumeran ciertos detalles que mejoran la usabilidad de la interfaz:

- Al seleccionar o deseleccionar los botones de selección de una opción u otra (controles numerados del 5 al 10 y el control 26 en la figura 4.7), las casillas donde se deben introducir los parámetros relativos a dicha opción se activan (permiten escribir en ellos) o se desactivan (sombreándose). De esta forma se le indica al usuario qué casillas debe rellenar.

- Al pulsar el botón 12, *comenzar*, antes de realizar las acciones indicadas por el usuario de la aplicación, mediante un cuadro de diálogo se avisa de la acción seleccionada y se pide confirmación al usuario. En las figuras 4.9, 4.10 y 4.11 se muestran ejemplos de estos cuadros de diálogo.

- Se muestran cuadros de progreso cuando se está realizando una ope_ra_ción larga, para evitar que dé la sensación de que la aplicación se ha quedado *colgada*. En las figuras 4.13, 4.14 puede verse dos ejemplos de cuadros de progreso.

- Cuando se guarda un fichero de configuración que contiene los pa_rá_me_tros especificados en la interfaz, y se le da el mismo nombre de otro archivo de configuración ya existente, se pide confirmación al usuario para reemplazar el archivo antiguo por el nuevo. Nótese que esto no ocurre con los ficheros de salida de la aplicación (red de neuronas, informe de errores, archivos de datos...), que se generan a partir del sufijo especificado en la casilla 1, `Fichero para salvar la sesión`, por lo que es importante asegurarse que no se van a sobrescribir archivos de anteriores experimentos antes de pulsar el botón 12, *comenzar*.

4.4.4. Algunos detalles de la implementación

En esta sección se detallan algunos aspectos de la implementación llevada a cabo para añadir las funcionalidades que se acceden desde la ventana de "Entrenamiento de redes".

Para permitir la aplicación de filtros a ficheros de datos, se ha generado una nueva clase que hereda de `DataProcessorLink`, denominada `ApplyFilterProcessor`. Esta clase representa un procesador que asume el rol de `Composite` (compuesto) en el patrón `Composite` que implementan todos los procesadores de datos que se utilizan en la aplicación MindReader (ver sección 4.1.3). Se han sobreescrito los métodos `initSession`, `processData` y `endSession` heredados de `DataProcessorLink` y se ha implementado el método `ApplyFilterProcessor::readFilterFromFile` que carga los datos del filtro desde el fichero.

El único constructor de `ApplyFilterProcessor` recibe como parámetros el nombre del fichero donde se especifica el filtro a aplicar y la dimensión de dicho filtro, así como un puntero al objeto `DataSession` que reciben el resto de procesadores. En primer lugar se invoca al constructor de la clase padre, pasándole como parámetro puntero al objeto `DataSession`. En dicho constructor se invoca a su vez a la clase padre de `DataProcessorLink`, `DataProcessor`, donde se inicializa el atributo `m_dataSession` con el puntero al objeto `DataSession`. Además se inicicializa a cero el atributo `m_processed`, que indica cuántos datos se han procesado hasta el momento. Una vez invocado a los constructores padres, los dos primeros parámetros del constructor `ApplyFilterProcessor` se utilizan para inicializar los atributos propios de la clase, `m_nameFileFilter`, `m_filterLength` y `m_filterWidth`.

El método `ApplyFilterProcessor::initSession` carga los datos del filtro del fichero cuyo nombre se pasó por parámetros en el constructor, invocando al metodo `ApplyFilterProcessor::readFilterFromFile`, y a continuación invoca a los métodos `initSession` de sus descendientes.

El método `ApplyFilterProcessor::processData` recibe un objeto de tipo `DataInstant`, que almacena la ristra de datos que se va a procesar, correspondiente a determinado instante de tiempo, y la longitud del array que almacena dichos datos. Al invocar a este método se aplica el filtro a los datos del objeto `DataInstant` recibido por parámetros, y el resultado se almacena en otro objeto `DataInstant` que se pasa como parámetro a los descendientes del procesador, a los que se invoca una vez aplicado el filtro. En la implementación concreta realizada, desde la clase `AutoLearnDlg` a este procesador sólo se le asigna un descendiente, de tipo `RawDataDumperProcessor`, que se encarga de escribir a fichero los resultados obtenidos.

La aplicación del filtro consiste en la multiplicación matricial de dicho filtro por el vector formado por los ocho valores correspondientes al valor de la señal en cada uno de los ocho canales del casco encefalográfico en determinado instante de tiempo (el objeto `DataInstant` recibido por parámetros). Por tanto, la longitud de ambos objetos `DataInstant` deberá ser la misma que la dimensión del filtro, es decir, el número de canales (ocho) utilizados del

casco electroencefalográfico.

Al invocar el método `ApplyFilterProcessor::endSession` se liberan recursos y se llama a los métodos `endSession` de los procesadores descendientes.

La clase `AutoLearnDlg`, que hereda de la clase perteneciente a la librería MFC `CDialog`, implementa la interfaz gráfica correspondiente a la ventana de "Entrenamiento de Redes" (ver figura 4.7) así como los métodos que se encargan de llevar la lógica para que se lleven a cabo las funcionalidades que permite dicha ventana. Al pulsar el botón "Comenzar" se genera un evento que hace que se ejecute el método `AutoLearnDlg::OnBnStart`. Desde dicho método se realizan ciertas comprobaciones de los parámetros facilitados por el usuario y según los botones de tipo radio que estén seleccionados se muestra el cuadro de diálogo de confirmación y se invocará a unos métodos u otros, que realizarán las acciones seleccionadas.

Estos métodos son los que se enumeran a continuación.

`AutoLearnDlg::simulateSession` invocado cuando se va a simular una sesión,

`AutoLearnDlg::trainNetWithPatternsFile` para entrenar una nueva red con los datos del PSD contenidos en uno o varios ficheros,

`AutoLearnDlg::trainOldNetWithPatternsFile` para reentrenar una red entrenada previamente con los datos del PSD contenidos en uno o varios ficheros,

`AutoLearnDlg::trainNetWithNewAdquisitionSessionData` para realizar una sesión de adquisición y luego entrenar una red de neuronas con los datos obtenidos en ella.

`ARFFGenerator::generate` para generar el archivo arff a partir de los datos del PSD contenidos en uno o varios ficheros.

Desde estos métodos se invocan a otros encargados de cargar los datos (como `AutoLearnDlg::loadPatterns`), se declaran y se inicializan los procesadores necesarios, asignando a cada uno los desdencientes requeridos para obtener la salida deseada en cada caso, y se invoca a los métodos implicados en el entrenamiento de redes de neuronas (`NeuralNetworkClassifier::train` o `NeuralNetworkClassifier::save`).

Según qué procesadores se hayan utilizado, la pila de llamadas a los procesadores será diferente, a continuación se comentan los diagramas de secuencia que representan la pila de llamadas realizada por la aplicación en cada caso.

La pila de llamadas que se realiza cuando se selecciona realizar una sesión de adquisición y aplicar un filtro a los datos en el dominio del tiempo se muestra en la figura 4.18. Si no se selecciona la opción de aplicar un filtro, el procesador la pila de llamadas es similar a la que se realizaba en la primera versión del programa, se puede consultar en la figura 4.3.

Si en lugar de realizar una nueva sesión de adquisición ésta es simulada a partir de los datos obtenidos en otra sesión y almacenados en un fichero de tiempo `_time.raw`, la pila de llamadas será la que se muestra en la figura 4.19 en caso de que se aplique un filtro a los datos y la de la figura 4.20 en caso de que no se aplique ningún filtro. En ambos diagramas se ve que ha desaparecido el primer procesador de tipo `RawDataDumperProcessor` que se utilizaba cuando se realizaba la sesión de adquisición (ver figuras 4.18 y 4.3), que volcaba los datos obtenidos del casco electroencefalográfico al fichero `_time.raw`.

Se puede observar cómo, para el caso del procesamiento de datos obtenidos a partir una simulación, no se emplean procesadores asíncronos, ya que en este caso no existen restricciones de tiempo real para ninguno del resto de procesadores, y el procesado puede hacerse secuencial. De esta forma el programa es mucho más eficiente que si se empleasen varios hilos para el procesado de datos.

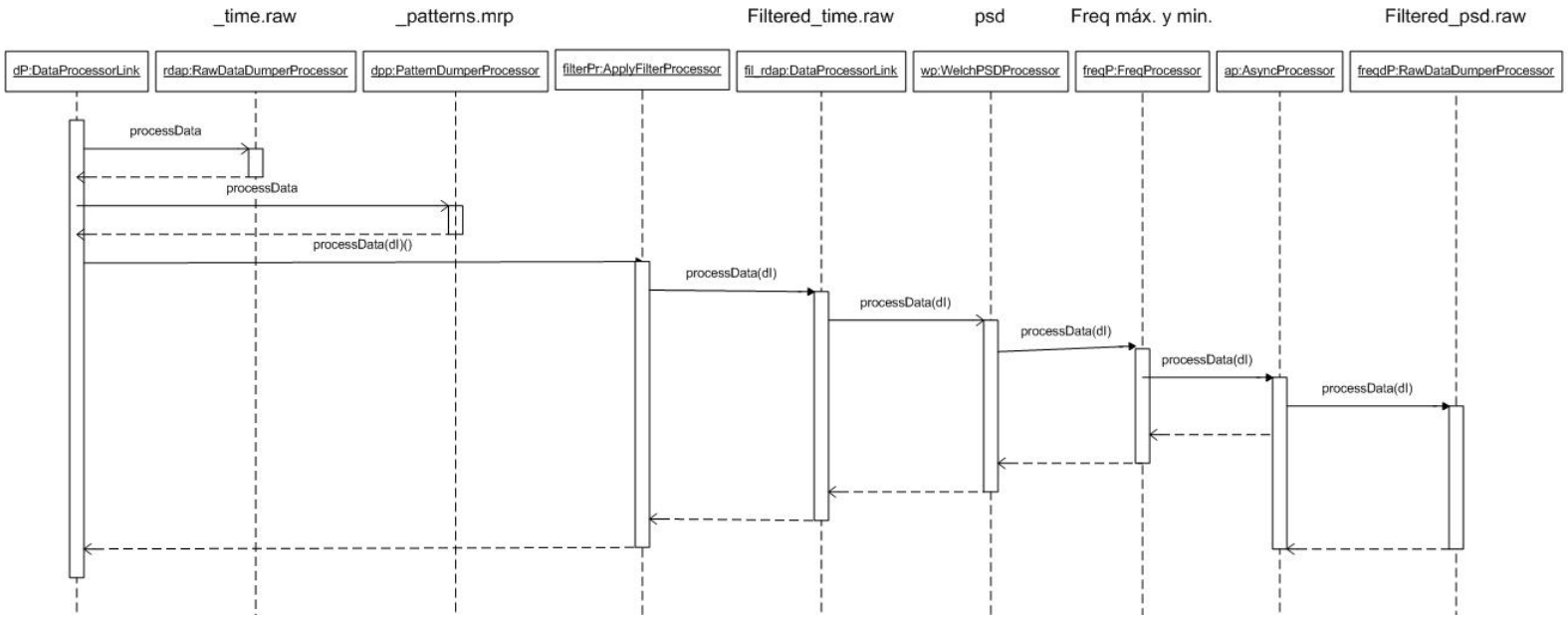

Figura 4.18: Diagrama de secuencia: Procesamiento de datos al realizar una nueva sesión de adquisición y al aplicar un filtro

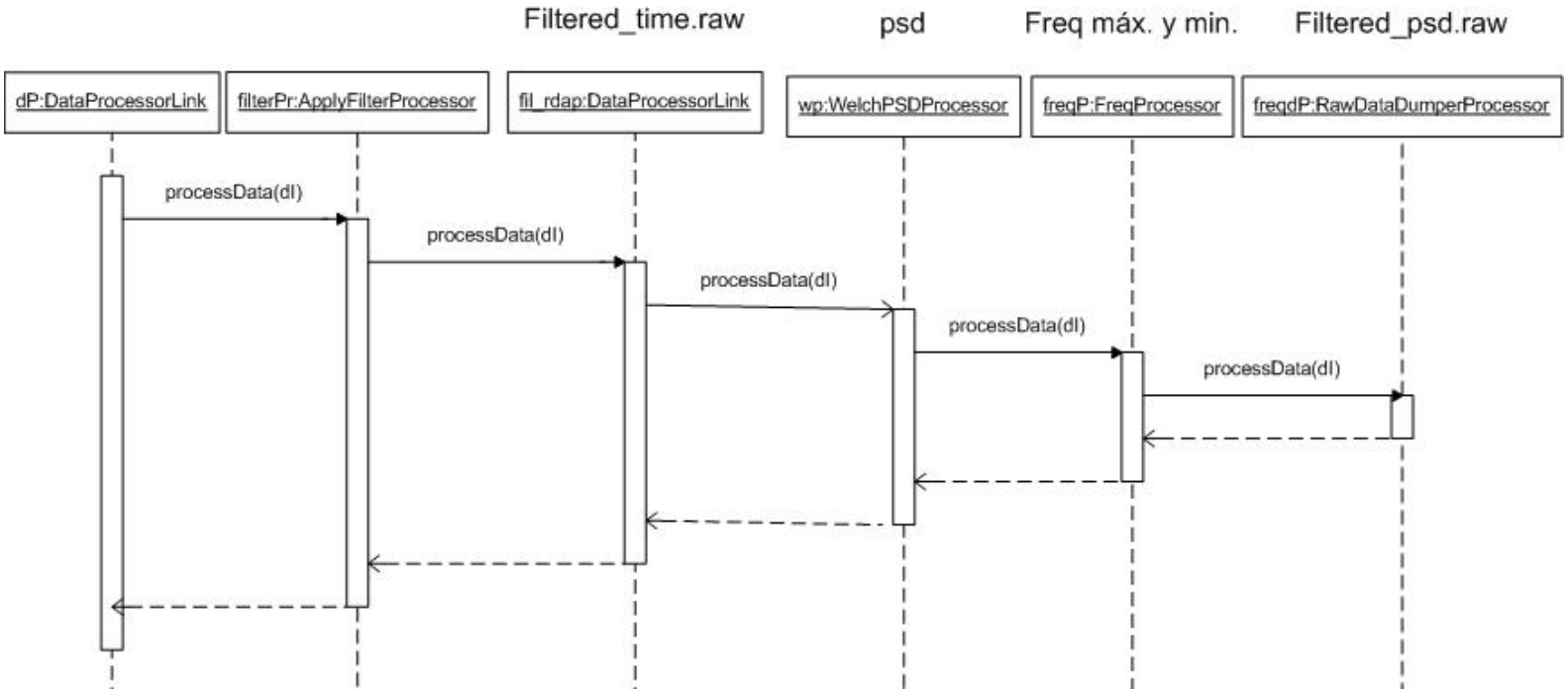

Figura 4.19: Diagrama de secuencia: Procesamiento de datos al simular una sesión de adquisición y aplicar un filtro

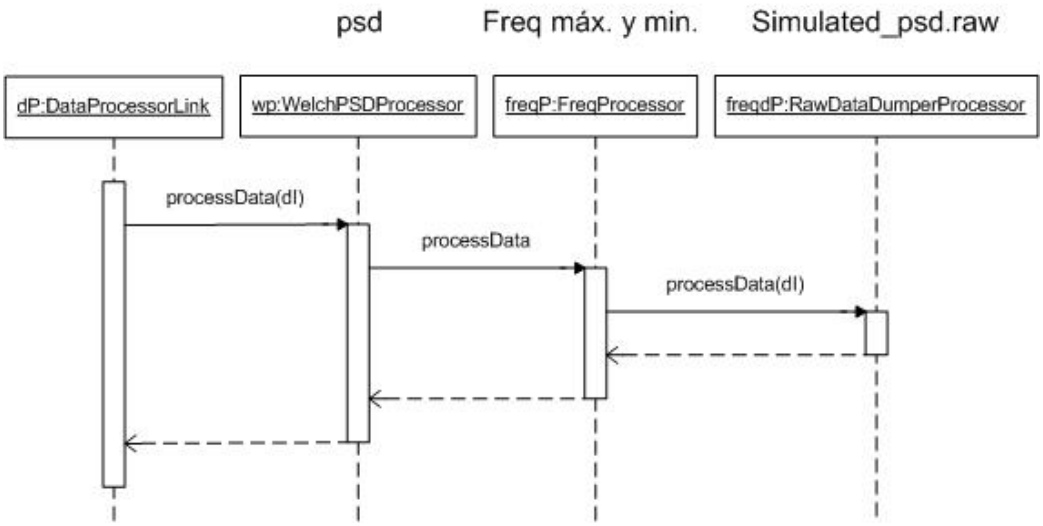

Figura 4.20: Diagrama de secuencia: Procesamiento de datos al simular una sesión de adquisición sin aplicar ningún filtro

Capítulo 5

Adquisición de datos

Una sesión de adquisición de datos consiste en colocar el casco encefalógrafo a una persona que se preste, y mediante la aplicación MindReader y el casco encefalógrafo junto su amplificador y el programa BrainVision Recorder, obtener señales encefalográficas clasificadas según lo que esté indicando el programa MindReader que debe pensar el sujeto en un determinado momento. En el capítulo 3 "Hardware y Software implicado"se ha explicado el funcionamiento de dichos elementos.

En una sesión de adquisición se realizan varios periodos de adquisición con descansos entre medias. Una vez realizada una sesión de adquisición, los datos obtenidos durante la misma deben procesarse y analizarse para poder extraer conclusiones.

En este capítulo se detallarán aquellos aspectos relativos a la adquisición de datos realizada durante el desarrollo de este proyecto de fin de carrera. Se comentarán los detalles de cada sesión de adquisición, las decisiones tomadas a la hora de procesar los datos y a la hora de analizarlos y por último, los resultados obtenidos en los análisis realizados.

5.1. Preparación de las sesiones de adquisición de datos

Antes de realizar un periodo de adquisición de datos se debe determinar qué parámetros se van a utilizar en el mismo.

Lo primero que debe decidirse es en qué clases va a pensar el sujeto durante la sesión. Se trata de acordar dos o tres clases con las que el sujeto se sienta cómodo en pensar en ellas. Como se ha comentado anteriormente en el capítulo 2, diferentes procesos mentales activan diferentes zonas del cerebro, o las activan de forma diferente. Además, para cierto tipo de procesos mentales

se conoce qué lóbulos del cerebro participan. Se trata de buscar pensamientos que activen de forma distinta una misma zona o que activen diferentes zonas del cerebro entre sí, de forma que un clasificador entrenado sea capaz de diferenciar si un patrón pertenece a un proceso mental o a otro. Se sugirieron varias clases a emplear, y los sujetos que realizaban una sesión de adquisición debían elegir entre ellas, previamente a realizar la sesión.

Las clases sugeridas fueron las siguientes:

- Realizar una multiplicación de varios dígitos mentalmente. De esta forma se pretende que se activasen las neuronas implicadas en el proceso de resolución de un problema aritmético. Se pidió al usuario que realizase multiplicaciones de varios dígitos para que el proceso mental realizado fuera más complejo y no se limitase a recordar la tabla de multiplicar.

- Rotar un objeto mentalmente (el sujeto se imaginaba un triángulo rotando sobre sí mismo).

- Relajarse.

- Mover mano derecha (tener la intención de moverla pero sin llegar a hacerlo).

- Mover mano izquierda (tener la intención de moverla pero sin llegar a hacerlo).

- Mover pie derecho (tener la intención de moverlo pero sin llegar a hacerlo).

- Mover pie izquierdo (tener la intención de moverlo pero sin llegar a hacerlo).

Otros parámetros que deben determinarse antes de realizar la sesión son la *duración de la clase* y la *duración de la pausa*. El programa MindReader indica al usuario en qué debe pensar en cada momento, mediante estímulos visuales o sonoros según la interfaz de sesión de adquisición que se haya escogido. Se indica que debe pensar en una clase determinada durante un periodo de tiempo, a continuación que debe pensar en la siguiente clase, y así sucesivamente hasta mostrar todas las clases de las que consta la sesión; entonces se vuelve a mostrar la primera clase y se continúa con la siguiente, etcétera, hasta que se pulsa el botón "terminar", momento en el que finaliza el periodo de adquisición de datos. A continuación podrían modificarse los parámetros especificados para que sean distintos en el siguiente periodo de

adquisición. Después de varios periodos de adquisición de datos se dará por terminada la sesión.

La duración del intervalo de tiempo que transcurre desde que se indica al usuario de la aplicación MindReader que se debe pensar en una clase determinada hasta que se indica que debe pensar en la siguiente, es igual para todas las clases y es la suma de los parámetros de *duración de la clase* y *duración de pausa*, que vienen dados en segundos. El parámetro *duración de la clase* es el tiempo "útil"de duración de la clase, durante el cual los datos recogidos se esperan emplear en los análisis. Si el programa debe realizar alguna acción cuando se ha alcanzado dicho tiempo (como avisar al usuario que está realizando una sesión de adquisición mediante la interfaz auditiva de que deje realizar un proceso mental y se prepare para recibir el estímulo sonoro del siguiente) se realizará al cabo de *duración de la clase* segundos, y a continuación se espera *tiempo de pausa* segundos antes de pasar a mostrar la siguiente clase o de emitir el sonido que indique el cambio.

Determinando las clases de las que consta la sesión y durante cuánto tiempo se debe pensar en ellas, el programa ya tiene la información básica necesaria para realizar una sesión de adquisición de datos. Puede etiquetar los datos recogidos a través del casco encefalógrafo como pertenecientes a la clase que se está indicando en un determinado momento. Para saber cuántas muestras de las recogidas pertenecen a cada clase, se necesita conocer otro parámetro: la *frecuencia de muestreo* del casco encefalográfico. La *frecuencia de muestreo* es el número muestras por segundo que recibe la aplicación MindReader del dispositivo electroencefalográfico, a través del servidor de datos Brain Vision Recorder. Está fijado a 500 Hz en el código de la aplicación MindReader, y no es configurable a través de la interfaz de la aplicación.

Para facilitar la labor al usuario del que se están adquiriendo datos, a cada clase se le asocia un color o un sonido. Este color o sonido, junto con un mensaje de texto, servirá para indicar al usuario en qué momento debe cambiar a pensar en la siguiente clase, y qué clase es ésta.

Estos son los datos imprescindibles para poder llevar a cabo una sesión de adquisición de datos que consista en capturar señales del casco, que se encuentran en el dominio del tiempo; y los etiquete para su posterior procesado y análisis. MindReader genera un fichero que contiene para cada instante el valor de la señal en el dominio del tiempo de cada uno de los canales del casco y otro fichero que indica la línea del fichero del dominio del tiempo donde comienza cada clase.

Además de generar dichos ficheros, realiza cierto procesamiento en los datos, de forma que genera otro archivo que contiene los datos en el dominio de la frecuencia. Para realizar esta transformación de la señal del dominio del tiempo al dominio de la frecuencia se necesita especificar ciertos parámetros:

el *Número de muestras para la Transformada Rápida de Fourier*, la *frecuencia máxima* y la *frecuencia mínima*.

Número de muestras para la Transformada Rápida de Fourier ("Fast Fourier Transform", FFT). Indica el tamaño de la ventana de datos en el dominio del tiempo a utilizar a la hora de calcular la FFT, es decir, el número de muestras usadas en el cálculo de la misma. La FFT es usada en el método de Welch para la estimación del PSD, para calcular el valor de la señal en el dominio de la frecuencia.

Frecuencia máxima y *frecuencia mínima*. Con los datos de la transformada rápida de Fourier se obtiene el valor de la señal en diferentes frecuencias, pero no todas son relevantes para nuestros propósitos. Los parámetros *frecuencia máxima* y *frecuencia mínima* determinan el subconjunto de frecuencias que se va a tomar para el procesado.

Con la funcionalidad de *Simulación de sesiones y entrenamiento de redes*, añadida a la aplicación durante este proyecto de fin de carrera, puede repetirse este procesado variando los parámetros, de forma que se pueden obtener distintos ficheros con los datos en el dominio de la frecuencia según los parámetros que se utilicen, sin necesidad de volver a realizar una sesión de adquisición propiamente dicha.

El parámetro *solapamiento* indica cuántos instantes de tiempo (o muestras) no se utilizan entre el cálculo de un patrón en el dominio de la frecuencia y el siguiente. Para calcular el PSD del instante n se necesitan las muestras desde la *n-n_muestras_fft* hasta la n, el siguiente patrón se calcularía con los datos de la muestra *(n+solapamiento)-n_muestras_fft* a la muestra *n+solapamiento*.

Además hay que ajustar el *número de patrones por clase* (pat_por_clase) y el *número de patrones a descartar* (pat_desc). Estos parámetros están directamente relacionados con los valores de *duración de clase* (t_clase) y *duración de pausa*(t_pausa) y con el *solapamiento* y *el número de muestras para la FFT* (n_muestras_FFT), y debe ajustarse según los valores de dichos parámetros. El parámetro *patrones por clase* (pat_por_clase) indica el número de patrones en el dominio de la frecuencia que se utilizarán de los generados cada vez que se muestra una clase durante *duración de la clase* (t_clase) segundos. En condiciones normales debería coincidir o ser inferior al número de patrones que se generan durante el tiempo de duración de la clase (t_clase). El número de patrones a descartar indica el número de patrones en el dominio de la frecuencia que se descartarán del inicio de cada clase, normalmente son los generados durante el tiempo de pausa (t_pausa). Se puede utilizar estas fórmulas para calcular los valores adecuados:

$$total\_num\_pat = (t\_clase + t\_pausa) * frec\_muestreo \tag{5.1}$$

$$pat\_desc = t\_pausa * frec\_muestreo \tag{5.2}$$

$$total\_num\_pat - pat\_desc >= pat\_por\_clase \tag{5.3}$$

$$pat\_por\_clase <= floor\left(\frac{t\_clase * frec\_muestreo - n\_muestras\_FFT)}{solapamiento}\right)+1 \tag{5.4}$$

Estos últimos parámetros se utilizan para construir el conjunto de patrones que se utiliza al entrenar los clasificadores. Bien se genera un fichero de patrones .arff para ser procesado con el programa WEKA o bien una estructura de datos (de tipo `DataSet`) que contenga los patrones si el clasificador se construye mediante el programa nnt o mediante la nueva funcionalidad de *simulación de sesiones* añadida a la aplicación MindReader durante el desarrollo del proyecto.

Además de los parámetros anteriormente citados, habrá que indicar el prefijo de los ficheros de salida que se van a generar. Se generará un fichero con los datos en el dominio del tiempo con el prefijo que se haya indicado y el sufijo `_time.raw`, otro con los datos en el dominio de la frecuencia con el sufijo `_psd.raw`, y el fichero con sufijo `_patterns.mrp` que indica a qué clases pertenecen los datos de los ficheros anteriores, donde también quedan reflejados los parámetros establecidos para el periodo de adquisición de datos. En la tabla resumen 5.1 se muestran todos los parámetros que hay que definir.

Parámetro	**Uso**
Nombre de clases a utilizar	Durante el periodo de adquisición.
Sonidos/colores asociados a cada clase	Durante el periodo de adquisición.
Duración de la clase (s)	Durante el periodo de adquisición.
Duración de la pausa (s)	Durante el periodo de adquisición.
Núm. muestras FFT	Cálculo del PSD con el método Welch.
Frec. máxima	Filtrado del PSD
Frec. mínima	Filtrado del PSD
Patrones generados por clase	Selección de patrones para la construcción del clasificador.
Patrones a descartar por clase	Selección de patrones para la construcción del clasificador.
Solapamiento	Selección de patrones para la construcción del clasificador.
Prefijo ficheros de salida	Cada vez que se genera un fichero de salida (Durante el periodo de adquisición, el cálculo del PSD y al generar un clasificador.)

Tabla 5.1: Parámetros que deben establecerse a la hora de realizar un periodo de adquisición de datos

5.2. Sesiones de adquisición realizadas

Una sesión de adquisición abarca una jornada de trabajo en la que se coloca el casco a un sujeto y se realizan varios periodos de adquisición de datos. Un periodo de adquisición de datos se inicia cuando se pulsa el botón "Comenzar"desde la ventana de configuración de la sesión, en ese momento se muestra la ventana de sesión de adquisición y se empieza a indicar al usuario que vaya realizando diferentes procesos mentales, y finaliza al pulsar el botón "Terminar" de dicha ventana, momento en el que se dejan de indicar más instrucciones al usuario y se vuelve a la ventana de configuración de la sesión. Entre un periodo de adquisición y otro se realiza un descanso de unos minutos de duración. En cada periodo se pueden utilizar parámetros diferentes, aunque conviene que se utilicen los mismos en varios periodos para que sean compatibles y se puedan agrupar para utilizarlos conjuntamente en la generación de clasificadores.

Han sido realizadas cinco sesiones de adquisición de datos en las que han participado tres voluntarios: David, Pablo y Marcos. En cada sesión de adquisición de datos que participaron se les colocó el casco encefalográfico y realizaron varios periodos de adquisición de datos de varios minutos de duración. Tres sesiones fueron realizadas por David, y Pablo y Marcos realizaron una única sesión cada uno. Estuvieron lo más quietos posible durante las sesiones y se intentó crear un ambiente de trabajo tranquilo para que no se despistasen durante las sesiones de adquisición.

En todas las sesiones de adquisición se mostraban las clases durante quince segundos cada vez, y se añadía un segundo más de tiempo de pausa. Ninguno de los sujetos pidió que se variara la duración del tiempo de clase. Es más, alguno comentó que un tiempo superior dificultaría la concentración.

Como frecuencia mínima y máxima se tomó 8 Hz y 30 Hz respectivamente, ya que frecuencias inferiores a 8 Hz son producto de los impulsos eléctricos provocados por el movimiento muscular del sujeto y superiores a 30 Hz se corresponden con las interferencias de la red eléctrica, por lo que incluirlas como entradas de los patrones sólo añadiría ruido.

También el valor de solapamiento se mantuvo constante a 31 patrones en todas las sesiones de adquisición, ya que en los estudios realizados por J. Millán se indica que es un valor adecuado.[5]

El número de muestras para la ventana de la transformada rápida de Fourier se mantuvo a 250 en todas las sesiones.

Los parámetros *patrones generados por clase* y *patrones a descartar por clase* que se establecieron no son relevantes, ya que fueron modificados a posteriori en todos los casos, alterando los valores que figuraban originalmente en los ficheros de sufijo `_patterns.mrp`.

A continuación se detallan los parámetros que varían entre las diferentes sesiones de adquisición. En las tablas 5.2 y 5.3 se muestra un resumen de todos estos datos.

5.2.1. David

La primera sesión de adquisición de David fue realizada el día 14 de Octubre del 2008, y constó de seis periodos de unos tres minutos de duración cada uno, con descansos entre medias. Las clases en las que eligió pensar fueron tres y se mantuvieron constantes a lo largo de los seis periodos de adquisición de datos: "realizar una multiplicación", "rotar un objeto" y "relajarse".

	1ª sesión David	**2ª sesión David**	**3ª sesión David**
Periodos	p1, p2, p3, p4, p5 y p6	p1, p2, p3, p4, p5, p6 y p7	p1, p2, p3, p4, p5, p6, p7, p8 y p9
pat solap	31	31	31
Duración clase (s)	15	15	15
Duración pausa (s)	1	1	1
Frec. máxima	30 Hz	30 Hz	30 Hz
Frec. mínima	8 Hz	8 Hz	8 Hz
Num muestras FFT	250	250	250
Clases	p1 a p6: 3 (multiplicar, rotar, relajarse)	p1 a p6 : 2 (mano dcha., mano izq) p7: 2 (mano dcha., rotar)	p1 y p3 a p9: 2 (mano dcha., mano izq.) p2: 2 (mano dcha., rotar)

Tabla 5.2: Parámetros usados en las sesiones de David

La segunda sesión de adquisición de datos realizada por David tuvo lugar el día 27 de Octubre del año 2008, y constó de siete periodos de entre dos y tres minutos de duración cada uno. En todos ellos se usaron las clases "mover mano derecha" y "mover mano izquierda"; menos en el último periodo, que hicimos una prueba con "rotar" y "mover mano derecha".

La tercera sesión de adquisición David constó de nueve periodos de unos dos minutos de recogida de datos, con descansos entre uno y otro. Esta sesión fue realizada el día 16 de Febrero del 2009. En todos los periodos de adquisición se emplearon las clases "mover mano derecha" y "mover mano izquierda", excepto en el segundo periodo de adquisición en el que se usó "rotar" y "mover mano derecha".

5.2.2. Pablo

La sesión de Pablo constó de 8 periodos de unos tres minutos cada uno, con descansos entre medias. Se realizó el mismo día en que se realizó la primera sesión de adquisición de David, el día 14 de Octubre del 2008. Con él se fueron variando en número y en tipo las clases en las que pensaba en los diferentes periodos de adquisición de datos. En los dos primeros periodos

eligió “mover mano derecha” y “rotar objeto”; en los tres siguientes periodos la clase “mover mano izquierda”se sustituyó por la clase “multiplicar”. En los tres últimos usó las mismas tres clases que David había usado en su primera sesión de adquisición, “multiplicar”, “rotar” y “relajarse”.

	Sesión Pablo	**Sesión Marcos**
Periodos	p1, p2, p3, p4, p5, p6, p7 y p8	p1, p2, p3, p4, p5 y p6
pat solap	31	31
Duración clase (s)	15	15
Duración pausa (s)	1	1
Frec. máxima	30 Hz	30 Hz
Frec. mínima	8 Hz	8 Hz
Num muestras FFT	250	250
Clases	p1 y p2: 2 (mano dcha., rotar) p3 a p5: 2 (multiplicar, rotar) p6 a p8: 3 (multiplicar, rotar, relajarse)	p1 a p6: 3 (mano dcha., mano izq., rotar)

Tabla 5.3: Parámetros usados en las sesiones de Pablo y Marcos

5.2.3. Marcos

Se realizó otra sesión de adquisición con un nuevo sujeto el día 20 de Marzo del 2009. La sesión de adquisición de Marcos constó de seis periodos de unos tres minutos cada uno y se emplearon las mismas tres clases en todos ellos: “mover mano dcha.”, “mover mano izqda.” y “rotar objeto”.

5.2.4. Sobre la colocación de los electrodos

En las distintas sesiones de adquisición no se colocaron los electrodos del casco electroencefalográfico exactamente igual en todas ellas. Se utilizaron los mismos ocho canales (F3, F4, FCz, Cz, C3, C4, T8 y T7) pero el orden de los mismos a la hora de conectarlos a los canales del amplificador (numerados del 1 al 8) difiere de una sesión a otra. Esto imposibilita el poder comparar los datos de unas sesiones con otras, utilizando los datos de una como conjunto de entrenamiento y los de otra como conjunto de validación.

Los electrodos fueron colocados como se muestra en la tabla 5.4

En la 1ª sesión de David, en la sesión de Pablo y en la sesión de Marcos sí que se realizaron con la misma configuración de los electrodos. Si los resultados de los clasificadores hubieran sido satisfactorios, se hubiera podido estudiar cómo clasificadores entrenados por determinados sujetos, y que funcionan de forma aceptable con ellos, se comportan con los datos de otras personas.

Sesión	Posiciones de los canales
1ª sesión David, sesión Pablo, sesión Marcos	1F3 2F4 3FCz 4Cz 5C4 6T8 7C3 8T7
2ª sesión David	1F3 2Cz 3F4 4T7 5C3 6Cz 7C4 8T8
3ª sesión David	1F3 2F4 3FCz 4T7 5C3 6Cz 7C4 8T8

Tabla 5.4: Colocación de los electrodos. Correspondencia canal amplificador - posición del electrodo en el casco

5.3. Análisis de los datos adquiridos

En los primeros análisis se procesaron los datos de cada periodo de forma independiente, utilizando un subconjunto de los datos de un mismo periodo para entrenamiento y otro para validación. Como se ha comentado anteriormente, dependiendo del valor del parámetro *solapamiento* los patrones pueden ser muy similares. Si se usan patrones muy parecidos para entrenar un clasificador se puede producir sobreaprendizaje y es menos probable que el clasificador pueda generalizar de forma adecuada.

Lo realmente interesante es ver cómo se comporta un clasificador que ha sido entrenado con los datos de varios periodos compatibles (aquellos correspondientes a periodos de adquisición realizados por la misma persona, y en el que los electrodos del casco electroencefalográfico estuviesen colocados de la misma forma y se hubiesen empleado las mismas clases) y validado con los datos de otro conjunto diferente de periodos compatibles con los anteriores. Para realizar este tipo de estudio, se han generado los ficheros `.arff` correspondientes a los conjuntos de entrenamiento y de validación y se utilizado la herramienta WEKA para realizar la experimentación con diferentes clasificadores. Los conjuntos de entrenamiento y de validación estarán compuestos de uno o varios periodos compatibles entre sí, siendo los periodos de entrenamiento y validación diferentes.

Mediante la herramienta WEKA pudo comprobarse el número de patrones de cada clase presentes en cada conjunto, y verificar que el porcentaje de cada clase era similar entre los conjuntos de entrenamiento y test; y que las clases estaban bien balanceadas, habiendo una representación equitativa de cada clase en los conjuntos. Por esta razón, puede asumirse que resultados de aciertos cercanos al 33,33 % a la hora de clasificar entre tres clases y al 50 % a la hora de hacerlo entre dos sería equivalente a una clasificación aleatoria.

Se realizaron diferentes análisis con los datos adquiridos, los parámetros que determinan qué patrones se van a utilizar para construir un clasificador varían entre ellos. En el caso del cuarto análisis además varía el número

de muestras para el cálculo de la FFT, lo que afecta al cálculo del PSD y por tanto al contenido del fichero que contiene los datos en el dominio de la frecuencia. Se van a comentar los parámetros empleados en cada uno de ellos.

En los primeros análisis realizados con los datos obtenidos de las dos primeras sesiones de adquisición de David y la sesión de Pablo se generó un fichero `.arff` por cada uno de los periodos realizados, sin agrupar periodos compatibles. Se descartaron 50 patrones por clase, que era el valor por defecto que asigna MindReader a dicho parámetro. De esta forma se descartaban tan sólo los datos adquiridos durante los 0.1 primeros segundos en que se muestra una clase al usuario. Aunque es un número diez veces inferior al que correspondería descartar los patrones generados durante el tiempo de pausa, no era descabellado comenzar descartando sólo unos pocos patrones para poder utilizar así mas patrones en la construcción del clasificador. En la tabla 5.5 se muestran los parámetros utilizados en el análisis que llamaremos a partir de ahora *Análisis 1*.

Parámetro	**Valor**
Patrones descartados inicio fichero	0
Patrones a descartar	50
Solapamiento	31
Patrones por clase	240
Duración de clase	15
Duración de pausa	1
Num muestras para FFT	250

Tabla 5.5: Parámetros *Análisis 1*: sobre los datos de las sesiones 1ª y 2ª de David y la sesión de Pablo

Se hizo un segundo análisis de los datos de la segunda sesión de adquisición de David, modificando el parámetro *patrones a descartar*. Esta vez se descartaron 500 patrones por clase, lo que se corresponde con descartar los datos adquiridos durante el primer segundo en que se muestra una clase, es decir, los generados durante el tiempo de pausa que se había establecido en un segundo.

El parámetro *patrones generados por clase* se ajustó para que fuese estric-

tamente el correspondiente a 15 segundos de duración de clase y 31 muestras de solapamiento aplicando la fórmula 5.4; es decir, 234 patrones por clase.

Además, en este análisis se descartaron los primeros 8000 datos de cada fichero, ya que los primeros valores que se obtienen en un periodo de adquisición no son fiables por varios motivos que se comentaron en la sección 4.2. Descartar una clase entera es muy simple, ya que sólo hay que comentar la línea correspondiente al inicio de dicha clase en el fichero de configuración de datos de la sesión `_patterns.mrp` antes de generar los archivos `.arff` o entrenar una red neuronal. En los análisis realizados a partir de éste, siempre se descartó la primera clase de cada fichero utilizado. En la tabla 5.6 se muestran los parámetros utilizados en este análisis (*Análisis 2*). Utilizando esta misma configuración se analizaron también los datos de la sesión de adquisición realizada por Marcos y de la tercera sesión de adquisición de David.

Parámetro	**Valor**
Patrones descartados inicio fichero	8000
Patrones a descartar	500
Solapamiento	31
Patrones por clase	234
Duración de clase	15
Duración de pausa	1
Num muestras para FFT	250

Tabla 5.6: Parámetros *Análisis 2*: sobre los datos de la sesiones 2ª y 3ª de David y la sesión de Marcos

Después de realizar estos análisis, se decidió aumentar todavía más el número de patrones a descartar del principio de cada clase. Cuando el usuario pasa de pensar en una clase a pensar en otra, el cambio de pensamiento no es inmediato y existe un periodo de tiempo en el que no está del todo concentrado en pensar ni lo uno ni lo otro, está adaptando su pensamiento a la nueva clase que se le muestra por pantalla, por esta razón se descartan estos patrones. Se decidió descartar dos segundos de datos, duplicando así el número de patrones descartados del inicio de cada clase, para ver si así mejoraban los resultados. Se modificó el parámetro patrones a descartar por clase a

1000 patrones, y se ajustó, aplicando la fórmula 5.4, el parámetro "patrones generados por clase" a 215, que son los que se generan en los 14 segundos "útiles" en los que se muestra la clase al usuario. Con esta nueva configuración (*Análisis 3*) que se muestra en la tabla 5.7 se volvieron a analizar los datos de las cinco sesiones de adquisición.

Parámetro	**Valor**
Patrones descartados inicio fichero	8000
Patrones a descartar	1000
Solapamiento	31
Patrones por clase	215
Duración de clase	14
Duración de pausa	2
Num muestras para FFT	250

Tabla 5.7: Parámetros Análisis 3: sobre los datos de todas las sesiones

Hasta el momento se habían usado en todos los análisis 250 muestras para el cálculo del PSD. Se decidió ampliar la ventana a 500 muestras, para comprobar si con ese valor los datos se comportaban mejor. Con 500 muestras se espera una mejor resolución de la señal en el dominio de la frecuencia, ya que al aumentar el tamaño de ventana al aplicar el método de Welch, se obtiene mayor resolución en el PSD. Se volvieron a procesar y analizar los datos de las cinco sesiones, generando los nuevos ficheros de datos en el dominio de la frecuencia y posteriormente indicando mediante el fichero `_patterns.mrp` que se descartasen 1000 patrones por clase, la primera clase de cada fichero y que se generasen 205 patrones por clase (este parámetro hubo de ajustarse al cambiar el número de muestras empleado en el cálculo de la FFT, mediante las ecuaciones 5.1, 5.2, 5.3 y 5.4). Los parámetros utilizados en este análisis (*Análisis 4*) se muestran en la tabla 5.8

A continuación se detallan los resultados obtenidos en el análisis de los datos de las sesiones de cada sujeto.

5.3.1. Datos de sesión de adquisición de Pablo

Como ya se comentó anteriormente en el apartado 5.2, Pablo realizó una única sesión de adquisición de adquisición con nueve periodos de adquisición

Parámetro	Valor
Patrones descartados inicio fichero	8000
Patrones a descartar	1000
Solapamiento	31
Patrones por clase	205
Duración de clase	14
Duración de pausa	2
Num muestras para FFT	500

Tabla 5.8: Parámetros *Análisis 4*: sobre los datos de todas las sesiones

de datos. Los parámetros empleados durante la misma se muestran en la tabla 5.3.

Se generaron las redes de neuronas y los archivos `.arff` a partir de los datos obtenidos en la sesión, empleando los parámetros del *Análisis 1* (tabla 5.5), del *Análisis 3* (tabla 5.7) y el *Análisis 4* (tabla 5.8). Para la generación de las redes de neuronas se utilizaron los parámetros siguientes: 20 *neuronas en las capas ocultas de la red*, *tasa de aprendizaje* 0,15, *momento de aprendizaje* 0,8, el 80 % de los datos fueron *utilizados para entrenamiento*, se realizaron 200 *ciclos de aprendizaje*y se generó un *informe con los errores de la red* cada 10 ciclos.

Para el *Análisis 1* no se agruparon los periodos compatibles, por lo que el único experimento que se realizó entrenando con un periodo y validando con otro fue usando los datos del periodo 1 con los del periodo 2. Del resto de periodos se obtuvieron redes neuronales entrenadas con un subconjunto de los datos del periodo y validadas con el resto mediante el programa nnt. Además se construyeron clasificadores de tipo MLP y SMO con validación cruzada de 3 y 5 subconjuntos respectivamente.

En el caso de los *Análisis 3 y 4* se agruparon los periodos compatibles, empleando el conjunto de datos formados por los datos de los periodos p5 y p6 como conjunto de entrenamiento, y el periodo p3 para realizar la validación. El conjunto de los datos de los periodos p7 y p8 se validó con los datos del periodo p9. El periodo 1 se usó como conjunto de entrenamiento que se validaba con los datos del periodo p2. Se construyeron clasificadores de tipo SMO utilizando los valores que asigna WEKA por defecto.

En la tabla 5.9 se muestra el número de patrones por clase en los conjuntos de entrenamiento (pat train) y de validación (pat test), así como el porcentaje de datos que pertenecen a cada clase en ambos conjuntos (cX train o cX test). En el caso del periodo p1 y p2, 'c0' es *mover mano derecha* y 'c1' es *rotar objeto*, en el caso de los periodos p3, p4 y p5, 'c0' es *realizar una multiplicación* y 'c1' es *rotar objeto*, por último, en el caso de los periodos p7, p8 y p9 'c0' es *realizar una multiplicación*, 'c1' es *rotar objeto* y 'c2' es *relajarse*. Se muestran estos valores para cada uno de los conjuntos de entrenamiento y validación que se especifican en la columna *Conjuntos* (donde los periodos que conforman el conjunto de validación van entre paréntesis) generados según los parámetros del análisis que se indica en la columna *Análisis*. Las clases están bien balanceadas y tienen un porcentaje similar de ocurrencias en los conjuntos de entrenamiento y de validación.

Conjuntos	Análisis	% c0 test	% c0 train	% c1 test	% c1 train	% c2 test	% c2 train	pat test	pat train
pablo s1 p7_8 (val p9)	Análisis 4	31,58	30,00	36,84	35,00	31,58	35,00	3895	4100
pablo s1 p7_8 (val p9)	Análisis 3	30,96	30,58	36,13	35,73	32,91	33,69	4166	4467
pablo s1 p5_6 (val p3)	Análisis 4	50,00	46,15	50,00	53,85	——-	——-	2050	5330
pablo s1 p5_6 (val p3)	Análisis 3	46,44	47,31	53,56	52,69	——-	——-	2315	5713
pablo s1 p1 (val p2)	Análisis 4	50,00	50,00	50,00	50,00	——-	——-	1640	2870
pablo s1 p1 (val p2)	Análisis 3	47,88	47,54	52,12	52,46	——-	——-	1796	3166
pablo s1 p1 (val p2)	Análisis 1	52,85	50,59	47,15	49,41	——-	——-	2176	3637

Tabla 5.9: Porcentaje de patrones de cada clase en los conjuntos utilizados en los análisis de los datos de la sesión de Pablo.

Empleando los parámetros del *Análisis 3* (tabla 5.7), descartando 1000 patrones por clase y la primera clase de cada periodo, generando 215 patrones por clase, y utilizando 250 muestras para el cálculo de la FFT, al validar un clasificador entrenado con el periodo p1 con el periodo p2, o viceversa, los resultados rondan el 50 % de aciertos. Lo mismo ocurre al validar un clasificador obtenido con los periodos p5 y p6 con el periodo p3.

Al validar un clasificador obtenido con los periodos p7 y p8 y validarlos con el p9, se obtiene tan sólo en torno al 31 % de aciertos. Estos porcentajes de aciertos son muy bajos ya que en los dos primeros casos se está clasificando entre dos clases y en el caso de los periodos p7, p8 y p9 entre tres clases.

En un análisis realizado anteriormente (*Análisis 1* -tabla 5.5) empleando los datos del periodo p1 para entrenar un clasificador SMO y como conjunto de validación los datos del periodo p2, pero descartando 50 patrones por clase en lugar de 1000, se obtenían resultados similares aunque ligeramente peores.

En el *Análisis 4* (tabla 5.8), que es similar al análisis *Análisis 3* pero utilizando 500 muestras para el cálculo de la FFT se obtienen una vez más resultados cercanos al 50 %, que incluso empeoran los resultados obtenidos en el análisis anterior.

En la tabla 5.10 se muestran estos resultados indicando qué periodos de datos se han usado para formar los conjuntos de entrenamiento y validación (columna *Conjuntos*), qué parámetros se han utilizado (los correspondientes al análisis que se indica en la columna *Análisis*), el clasificador empleado (en los resultados que se muestran siempre se ha usado el clasificador SMO), el porcentaje de aciertos obtenido al clasificar el conjunto de validación y el número de clases entre las que se debe clasificar.

Conjuntos	Análisis	Clasificador	% aciertos	Num clases
pablo s1 p7_8 (val p9)	Análisis 4	SMO	31,84	3
pablo s1 p7_8 (val p9)	Análisis 3	SMO	31,69	3
pablo s1 p5_6 (val p3)	Análisis 4	SMO	51,07	2
pablo s1 p5_6 (val p3)	Análisis 3	SMO	51,14	2
pablo s1 p1 (val p2)	Análisis 4	SMO	45,98	2
pablo s1 p1 (val p2)	Análisis 3	SMO	52,06	2
pablo s1 p1 (val p2)	Análisis 1	SMO	51,30	2

Tabla 5.10: Resultados obtenidos en los análisis de los datos de la sesión de Pablo.

Podemos concluir que con los datos de la sesión de Pablo al generar clasificadores entrenados con varios periodos y utilizando otro periodo como conjunto de test no se obtienen buenos resultados. El clasificador es incapaz de clasificar correctamente las clases del conjunto de validación independientemente del procesamiento de datos realizados. Tampoco los clasificadores entrenados y validados con los datos pertenecientes a un mismo periodo obtenían buenos resultados.

5.3.2. Datos de la primera sesión de adquisición de David

David realizó tres sesiones de adquisición, en este apartado se van a comentar los resultados obtenidos al analizar los datos de la primera de ellas. Como ya se comentó en el apartado 5.2, en su primera sesión de adquisición realizó seis periodos de adquisición de datos durante los cuales se concentró en los mismos procesos mentales: *multiplicar*,*rotar* y *relajarse*. Los parámetros empleados durante la misma se muestran en la tabla 5.2.

Con los datos obtenidos durante esta sesión se generaron los archivos `.arff` de los conjuntos de entrenamiento y validación empleando los parámetros de los análisis *Análisis 1* (tabla 5.5), *Análisis 2* (tabla 5.6), *Análisis 3* (tabla 5.7) y *Análisis 4* (tabla 5.8).

En el caso del *Análisis 1* sólo se generaron los archivos correspondientes a cada periodo, y uno que agrupaba los datos de todos los periodos. Se analizaron los datos validándolos consigo mismos, entrenando clasificadores de tipo SMO y MLP con validación cruzada. Los resultados de este análisis no son representativos al haber sido validados consigo mismos.

Al utilizar los parámetros del *Análisis 2* y del *Análisis 3* se empleó como conjunto de entrenamiento los datos de los periodos p1 a p5, y como conjunto de validación los datos del periodo p6. Por último se utilizaron los parámetros del *Análisis 4* para generar un archivo con los datos de los periodos p1 a p4, que se usó como conjunto de entrenamiento y otro con los datos de los periodos p5 a p6, que se usó como conjunto de validación. Se generaron clasificadores de tipo SMO y en el caso del *Análisis 3* además se generó un clasificador de tipo MLP.

En la tabla 5.11 se muestra el porcentaje de patrones de cada clase presentes en los conjuntos de validación y entrenamiento empleados en los análisis expuestos. Se llama 'c0' al proceso mental *multiplicar*,'c1' a *rotar* y 'c2' a *relajarse*. Se observa que las clases están bien balanceadas y que los porcentajes de cada clase en los conjuntos de validación y de entrenamiento son similares.

Conjuntos	Análisis	% c0 test	% c0 train	% c1 test	% c1 train	% c2 test	% c2 train	pat test	pat train
david s1 p1_2_3_4 val(p5y6)	Análisis 4	25,00	25,00	37,50	37,50	37,50	37,50	3280	6560
david s1 p1_2_3_4_5 val (p6)	Análisis 3	27,27	25,64	36,36	37,18	36,36	37,18	2365	8096
david s1 p1_2_3_4_5 val (p6)	Análisis 2	33,93	35,24	33,04	32,38	33,04	32,38	2096	10377

Tabla 5.11: Porcentaje de patrones de cada clase en los conjuntos utilizados en los análisis de los datos de la sesión de la 1ª sesión de David.

En la tabla 5.12 se muestra un resumen de los resultados obtenidos en los experimentos mencionados, indicando los parámetros utilizados (los correspondientes con los del análisis de la columna *Análisis*), el clasificador empleado, el porcentaje de aciertos obtenidos al clasificar el conjunto de test y el número de clases entre las que se debe clasificar.

Conjuntos	Análisis	Clasificador	% aciertos	Num clases
david s1 p1_2_3_4 val(p5 y p6)	Análisis 4	SMO	52,68	3
david s1 p1_2_3_4_5 val (p6)	Análisis 3	SMO	53,07	3
david s1 p1_2_3_4_5 val (p6)	Análisis 3	MLP	45,98	3
david s1 p1_2_3_4_5 val (p6)	Análisis 2	SMO	35,24	3

Tabla 5.12: Resultados obtenidos en los análisis de los datos de la 1ª sesión de David.

Lo más importante de todo es que descartando 1000 patrones del inicio de cada clase y aplicando SMO o MLP utilizando como conjunto de entrenamiento los datos de los periodos p1, p2, p3, p4, p5 y validando dichos clasificadores con los patrones obtenidos en el periodo p6 (*Análisis 3*) se obtienen en torno a un 53 % de aciertos al generar un clasificador SMO, lo que no está tan mal clasificando tres clases. Si se genera un clasificador de tipo MLP el resultado es algo peor. Cuándo sólo se descartaban la mitad de patrones del inicio de cada clase (500), como ocurre en el *Análisis 2*, los resultados eran significativamente peores, un 35,24 %.

Si se cambia el número de muestras para la FFT a 500 (*Análisis 4*) y se entrena con los periodos p1 a p4 y se valida con el conjunto formado por los datos de los periodos p5 y p6, se obtiene un 52,68 % de aciertos, resultado muy similar al obtenido en el *Análisis 3*. Aunque estos resultados son aceptables, un porcentaje cercano al 50 % de aciertos no es suficiente para utilizarlo en un sistema BCI que permita al usuario comunicarse satisfactoriamente con una máquina.

Aunque los porcentajes de aciertos no son representativos para valorar su uso en el sistema BCI, se muestra a modo ilustrativo en la tabla 5.13 la evolución del número de aciertos obtenidos por clasificadores SMO generados durante los análisis, al utilizar como conjunto de entrenamiento el conjunto formado por los datos de los periodos p1 a p6 de la primera sesión de David y empleando validación cruzada de 5 subconjuntos, con vistas a determinar la influencia de los parámetros empleados en los diferentes análisis.

Conjunto	**Análisis 1**	**Análisis 3**	**Análisis 4**
david s1 p1_2_3_4_5_6	34,96	54,28	68,36

Tabla 5.13: Porcentaje de aciertos obtenidos en los análisis de los datos de la 1ª sesión de David utilizando un clasificador SMO y validación cruzada de cinco conjuntos.

Mediante este ejemplo concreto se puede apreciar la evolución favorable de los resultados en los diferentes análisis, según los parámetros varían. No se dispone del porcentaje de aciertos relativo al segundo análisis (donde se descartan 500 muestras del inicio de cada clase) ya que no se realizó dicha prueba. Se aprecia cómo en cada análisis realizado el porcentaje de aciertos va aumentando, lo que nos indica que el ajuste de parámetros realizado en cada nuevo análisis no andaba muy desencaminado, aunque los resultados no sean todo lo buenos que se podría desear.

5.3.3. Datos de la segunda sesión de David

En este apartado se comentan los resultados obtenidos al analizar los datos de la segunda sesión de adquisición de David. Esta sesión, como se comentó en la sección 5.2, constó de siete periodos de adquisición durante los cuales se emplearon las mismas dos clases (*mover mano derecha* y *mover mano izquierda*) excepto en el último que se sustituyó la clase *mover mano izquierda* por *rotar*.

Se generaron los archivos *arff* de cada uno de los periodos empleando los parámetros de los análisis *Análisis 1* (tabla 5.5) y *Análisis 2* (tabla 5.6). Los ficheros obtenidos se emplearon para entrenar varios clasificadores en WEKA (SMO y MLP principalmente) con validación cruzada.

Posteriormente se generó, empleando los parámetros del *Análisis 3* (tabla 5.7), el conjunto formado con los datos de los periodos p1 a p5 y un archivo que contenía los datos del periodo p6. El primer conjunto se usó como conjunto de entrenamiento y el segundo como conjunto de validación para construir un clasificador SMO y un clasificador de tipo MLP.

Por último, se creó el archivo que contenía los datos de los periodos p1 a p4, generados con los parámetros del *Análisis 4* (tabla 5.8). Al igual que en el caso anterior, el conjunto con mayor número de patrones se utilizó para entrenar el clasificador y el más pequeño como conjunto de validación. Se generó un clasificador SMO. Además se generaron dos redes neuronales de la librería FANN empleando cada conjunto por separado de forma que se empleaba un porcentaje de datos de un conjunto para entrenamiento y el resto para validación. Para realizar el entrenamiento se emplearon los mismos parámetros que se emplearon a la hora de analizar los datos de Pablo, especificados en el apartado 5.3.1, con la salvedad de que se realizaron 500 ciclos de entrenamiento en lugar de 200.

En la tabla 5.14 se muestra el porcentaje de patrones de cada clase presentes en los conjuntos de validación y entrenamiento empleados en los análisis expuestos. En esta tabla se hace referencia a la clase *mover mano derecha* como 'c0' y *mover mano izquierda* como 'c1'. Se aprecia que las clases están representadas de forma equilibrada en cada conjunto, estando bien balanceadas y con una distribución similar en los conjuntos de entrenamiento y validación.

En la tabla 5.15 se muestra un resumen de los resultados obtenidos. Las columnas de la tabla son similares a las tablas 5.10 y 5.12.

Los resultados obtenidos al analizar los datos de esta sesión vuelven a ser desfavorables, obteniendo resultados cercanos al 50 % de aciertos independientemente del procesado que se realice a los datos. Al entrenar clasificadores con los datos de los periodos p1 a p5 y validarlos con el periodo p6,

Conjuntos	Análisis	% c0 test	% c0 train	% c1 test	% c1 train	pat test	pat train
david s2 p1_2_3_4 val(p5_6)	Análisis 4	50,00	48,00	50,00	52,00	4100	5125
david david s2 p1_2_3_4_5 val(p6)	Análisis 3	50,00	45,03	50,00	54,97	2580	7666

Tabla 5.14: Porcentaje de patrones de cada clase en los conjuntos utilizados en los análisis de los datos de la sesión de la 2ª sesión de David.

Conjuntos	Análisis	Clasificador	% aciertos	Num clases
david s2 p1_2_3_4 val(p5_6)	Análisis 4	SMO	52,76	2
david s2 p1_2_3_4_5 val(p6)	Análisis 3	SMO	51,12	2
david s2 p1_2_3_4_5 val(p6)	Análisis 3	MLP	51,09	2

Tabla 5.15: Resultados obtenidos en los análisis de los datos de la 2ª sesión de David.

habiendo descartado 1000 patrones por clase y 250 muestras para el cálculo de la FFT (*Análisis 3*) se obtiene un 51,12 % de aciertos. El número de aciertos es muy similar tanto si se utilizan 250 muestras para el cálculo de la FFT como si se usan 500 (*Análisis 4*): al validar con los datos de los periodos p5 y p6 un clasificador entrenado con los datos de los periodos p1 a p4, se obtiene un 52,76 %.

No obstante, al observar los porcentajes de aciertos obtenidos al analizar los datos de los conjuntos mencionados anteriormente empleando SMO validando con validación cruzada de 5 conjuntos, o con el programa nnt que genera redes de la librería FANN, se verifica que el hecho de utilizar 500 muestras para el cálculo de la FFT en lugar de 250 es positivo. Al utilizar 250 muestras se obtienen clasificadores que aciertan en torno a un 60 % de las veces, mientras que cuando se usan 500 muestras ese porcentaje sube hasta más del 80 % de aciertos. El número de aciertos elevado es normal dado que existe solapamiento entre los patrones que se encuentran en los conjuntos de entrenamiento y validación, lo que significa que hay patrones muy similares que se encontrán en ambos conjuntos.

Conjunto	Train err	Test err	% aciertos	% err c0	% err c1
david s2 p1_2_3_4	0,0265	0,0529	81,42	18,12	19,02
david s2 p5_6	0,0263	0,0502	82,83	14,78	19,40

Tabla 5.16: Resultados obtenidos al generar redes de neuronas FANN con los datos de la segunda sesión de David empleando los parámetros del Análisis 4

En la tabla 5.16 se muestra el tanto por uno de errores obtenido en el últi-

mo ciclo en entrenamiento (train err) y en validación (test err), el porcentaje de aciertos (% aciertos) al clasificar el conjunto de validación y el porcentaje de errores cometidos al clasificar la clase *mover mano derecha* (err c0) y al clasificar la clase *mover mano izquierda* (% err c1). Los resultados que se obtienen al utilizar datos de un mismo fichero para validar y entrenar son mucho mejores en contraposición con los obtenidos al entrenar con los datos de unos periodos y validar con los de otros (ver tablas 5.15 y 5.16), debido al solapamiento que hay entre patrones de ambos conjuntos en el primer caso.

5.3.4. Datos de la tercera sesión de David

En este apartado se comentan los resultados obtenidos al analizar los datos de la tercera sesión de adquisición de David, que constó de nueve periodos periodos de adquisición. En la sección 5.2 se especificaron los detalles de la misma, en la tabla 5.2 se muestran los parámetros empleados.

Se generaron las redes de neuronas y los archivos `.arff` a partir de los datos obtenidos en la sesión, empleando los parámetros del *Análisis 2* (tabla 5.6), del *Análisis 3* (tabla 5.7) y el *Análisis 4* (tabla 5.8). Para la generación de las redes de neuronas se utilizaron los mismos parámetros de entrenamiento empleados en los análisis de otras sesiones, especificados en el apartado 5.3.1.

En primer lugar se generaron dos archivos *arff* con los datos de varios periodos, uno con los de los periodos p8 y p9 y otro con los de los periodos p1, p3, p4, p5, p6 y p7 empleando los parámetros del *Análisis 2*. Se generó un clasificador SMO empleando como conjunto de entrenamiento el conjunto de mayor número de patrones y como conjunto de validación el menor.

Posteriormente se probó a descartar más patrones por clase, empleando los datos del *Análisis 3*. Se generó el archivo `.arff` que contiene los datos de los periodos p3 a p7, que se usó como conjunto de entrenamiento para validar otro clasificador SMO. Como conjunto de validación se emplearon los datos del periodo p1.

Por último, se probó a simular una sesión de adquisición a partir de los datos de los periodos p1, p3, p4, p5, p6 y p7 empleando esta vez 500 muestras para el cálculo de la FFT y se simuló de forma análoga una sesión de adquisición con los datos de los periodos p8 y p9. Se generaron de esta forma los archivos `.arff` de dichos conjuntos según los parámetros del *Análisis 4* y se generó un clasificador SMO empleando el primero como conjunto de entrenamiento y el segundo como conjunto de validación.

En la tabla 5.17 se muestra el porcentaje de patrones de cada clase presentes en los conjuntos de validación y entrenamiento empleados en los análisis expuestos. En esta tabla *mover mano derecha* se corresponde con la clase

'c0' y *mover mano izquierda* se corresponde con la clase 'c1' Se observa que el porcentaje de cada clase en los conjuntos de validación y entrenamiento son similares y que las clases están bien balanceadas.

Conjuntos	Análisis	% c0 test	% c0 train	% c1 test	% c1 train	pat test	pat train
david s3 p1_3_4_5_6_7 val(p8_9)	Análisis 4	45,00	48,33	55,00	51,67	4100	12300
david s3 p3_4_5_6_7 val(p1)	Análisis 3	47,44	47,89	52,56	52,11	2266	15593

Tabla 5.17: Porcentaje de patrones de cada clase en los conjuntos utilizados en los análisis de los datos de la sesión de la 3ª sesión de David.

En la tabla 5.18 se muestra un resumen de los resultados obtenidos. Las columnas de la tabla son similares a las de las tablas 5.10, 5.12 y 5.15 .

Conjuntos	Análisis	Clasificador	% aciertos	Num clases
david s3 p1_3_4_5_6_7 val(p8_9)	Análisis 4	SMO	54,83	2
david s3 p3_4_5_6_7 val(p1)	Análisis 3	SMO	57,11	2
david s3 p3_4_5_6_7 val(p1)	Análisis 3	MLP	53,66	2
david s3 p1_3_4_5_6_7 val(p8_9)	Análisis 2	SMO	55,41	2
david s3 p1_3_4_5_6_7 val(p8_9)	Análisis 2	MLP	53,14	2

Tabla 5.18: Resultados obtenidos en los análisis de los datos de la 3ª sesión de David.

Al utilizar como conjunto de entrenamiento los datos de los periodos p1, p3, p4, p5, p6 y p7 y como conjunto de validación los de los periodos p8 y p9 descartando 500 patrones por clase y se empleando 250 muestras en el cálculo de la FFT (*Análisis 2*) se obtuvo un 55.41 % de aciertos, lo que vuelve a ser unos resultados de aciertos muy pobres. De forma análoga, clasificando los datos con MLP se obtiene un 53.14 % de aciertos.

Al descartar las 1000 primeras muestras de cada clase (*Análisis 3*) y utilizar un clasificador SMO empleando como conjunto de test los datos del periodo p1 y como conjunto de entrenamiento los datos de los periodos p3 a p9, se obtiene un 57,11 % de aciertos, con MLP obtiene un 53,66 % , lo cual no son buenos resultados.

Al utilizar 500 muestras para el cálculo de la FFT (*Análisis 4*), el porcentaje de aciertos obtenido asciende al 54,83 %.

Al igual que en casos anteriores, los resultados de las redes de neuronas generadas (donde se utiliza un conjunto de validación con datos que probablemente sean bastante similares a los que contiene el conjunto de entrenamiento debido al solapamiento entre patrones) parecen ser mejores cuando se emplean 500 muestras para el cálculo de la FFT que cuando se utilizan 250.

Además de estos análisis se realizaron otros experimentos empleando los parámetros del *Análisis 2*. Los resultados que se muestran en la tabla 5.18 fueron obtenidos utilizando ficheros *arff* generados por el programa nnt, que normaliza automáticamente por atributos los patrones. Se habían generado dos ficheros `.arff` que contenían los datos de los de los periodos p8 y p9 uno y el otro los de los periodos p1, p3, p4, p5, p6 y p7. Se descartaron 500 patrones por clase y se empleaban 250 muestras en el cálculo de la FFT. Se generaron además los archivos análogos a los anteriores con el script para Octave, obteniendo así los datos sin normalizar y ordenados, para poder realizar diferentes procesados antes de analizar los datos. Se procesaron con SMO en WEKA volviendo a utilizar el conjunto formado por los datos de los periodos p8 y p9 como conjunto de test y el otro conjunto como conjunto de entrenamiento. Con el programa WEKA se probó a analizar los datos aleatorizándolos y normalizándolos previamente, obteniendo los resultados presentados en la tabla 5.19, que no mejoran los anteriormente mencionados.

Sesión 3ª de David: p1, p3 a p7 (val p8 y p9	**% Aciertos**
Datos sin normalizar y ordenados (tal cual están)	51.52 %
Datos normalizados por atributos	52.63 %
Datos normalizados por instancias	55.17 %
Datos desordenados (aplicando el filtro de weka para aleatorizar)	51.52 %
Datos desordenados y normalizados por atributos	52.63 %
Datos desordenados y normalizados por instancias	55.12 %

Tabla 5.19: Resultados obtenidos en los análisis de los datos de la 3ª sesión de David preprocesando los datos con WEKA

5.3.5. Datos de la sesión de adquisición de Marcos

Marcos realizó una única sesión de adquisición, los resultados obtenidos al analizar estos datos se comentan en este apartado. En la sección 5.2 se especifican los detalles de esta sesión, en la cual se realizaron seis periodos de adquisición en los que se emplearon las mismas tres clases en todos ellos. En la tabla 5.3 se especifican los parámetros empleados en la misma.

Los datos se analizaron empleando los parámetros del *Análisis 2* (tabla 5.6), del *Análisis 3* (tabla 5.7) y el *Análisis 4*.

Con los parámetros del *Análisis 2* sólo se generaron los archivos correspondientes a cada periodo y se generaron redes de neuronas entrenándolas empleando un porcentaje de los datos de cada periodo como conjunto de entrenamiento y el resto para validar. En el entrenamiento se emplearon los

mismos parámetros de entrenamiento empleados en los análisis de otras sesiones, especificados en el apartado 5.3.1.

Con los parámetros del *Análisis 3* se generó el archivo `.arff` que contenía los datos de los periodos p1, p2, p3, p4, p6 y otro archivo `.arff` con los datos del periodo p5. El primero se usó para entrenar un clasificador SMO y el segundo para validarlo. De forma análoga se generó un clasificador MLP.

Por último, con los parámetros del *Análisis 4* se generaron los archivos `.arff` con los periodos p1 a p4 (empleado como conjunto de entrenamiento) y con los datos de los periodos p5 y p6 (empleado como conjunto de validación). Se construyó un clasificador SMO.

En la tabla 5.20 se muestra el porcentaje de patrones de cada clase presentes en los conjuntos de validación y entrenamiento empleados en los análisis expuestos, las clases están bien balanceadas y el porcentaje de cada clase es similar en los conjuntos de entrenamiento y de validación. La clase *mover mano derecha* se corresponde con la etiqueta 'c0', la clase *mover mano izquierda* se corresponde con la etiqueta 'c1' y la clase *rotar* se corresponde con la etiqueta 'c2'.

Conjuntos	Análisis	% c0 test	% c0 train	% c1 test	% c1 train	% c2 test	% c2 train	pat test	pat train
marcos s1 p1_2_3_4 val (p5_6)	Análisis 4	33,33	31,02	33,33	35,45	33,33	33,24	4920	9252
marcos s1 p1_2_3_4_6 val (p5)	Análisis 3	32,76	32,76	34,48	35,10	32,76	32,91	2625	12412

Tabla 5.20: Porcentaje de patrones de cada clase en los conjuntos utilizados en los análisis de los datos de la sesión de Marcos.

En la tabla 5.21 se muestra un resumen de los resultados obtenidos. Las columnas de esta tabla siguen la misma estructura que las de las tablas 5.10, 5.12, 5.15 y 5.18 .

Conjuntos	Análisis	Clasificador	% aciertos	Num clases
marcos s1 p1_2_3_4 val (p5_6)	Análisis 4	SMO	33,68	3
marcos s1 p1_2_3_4_6 val (p5)	Análisis 3	SMO	36,00	3
marcos s1 p1_2_3_4_6 val (p5)	Análisis 3	MLP	31,31	3

Tabla 5.21: Resultados obtenidos en los análisis de los datos de la sesión de Marcos.

Al descartar 1000 muestras de cada clase y empleando 250 muestras para la FFT (*Análisis 3*) se vuelven a obtener malos resultados (31,31 % de aciertos con MLP y un 36 % con SMO, cercanos al 33.33 % que indica que el clasificador se comporta como un clasificador aleatorio).

Al emplear 500 muestras en el cálculo de la FFT (*Análisis 4*), los resultados siguen siendo muy malos, obteniendo un 33,68 % de aciertos al aplicar

Conjunto	Train err	Test err	% aciertos	% err c0	% err c1	% err c2
marcos s1 p1_2_3_4	0,1340	0,1463	57,29	44,80	43,52	40,41
marcos s1 p5_6	0,0787	0,0952	69,27	42,62	27,34	21,93

Tabla 5.22: Resultados obtenidos al generar redes de neuronas FANN con los datos de la sesión de Marcos empleando los parámetros del Análisis 4

SMO. Pero si se comparan los resultados que se obtienen de las redes entrenadas con el programa nnt con los obtenidos anteriormente en el resto de los análisis, sí que parece que es más apropiado emplear 500 muestras para el cálculo de la FFT, ya que los porcentajes de aciertos alcanzan cotas superiores al 55 % donde antes obtenían valores en torno al 33 % de aciertos. En la tabla 5.22 se muestran los resultado obtenidos por las redes de neuronas de la libería FANN, los resultados son mejores que los que se obtienen al entrenar y validar con conjuntos distintos, ya que, como se ha comentado, existe solapamiento entre los patrones de ambos conjuntos.

5.3.6. Discusión de los resultados

En general, el porcentaje de aciertos de los clasificadores obtenidos ronda el 50 % cuando se clasifica entre dos clases y el 33,33 % cuando es entre tres clases, independientemente de si se han descartado 50, 500 ó 1000 patrones por clase, o si se ha descartado la primera clase de cada fichero, o de si el número de muestras para la FFT son 250 patrones ó 500... (Ver tablas 5.10, 5.15, 5.18 y 5.21).

En el único caso en el que se obtienen resultados relativamente aceptables al entrenar con unos periodos y validar con otros, es al analizar los datos de la sesión 1 de David (tabla 5.12). Se obtienen resultados de aciertos superiores al 50 % al clasificar patrones que pueden pertenecer a tres clases. No obstante éste sigue siendo un porcentaje de acierto muy bajo que no permitiría utilizar el clasificador generado para formar parte de un sistema BCI.

No se observa una variación significativa en el porcentaje de aciertos que obtiene un clasificador si se normalizan y/o aleatorizan los datos previamente a su utilización en la construcción del mismo (ver tabla 5.19).

No obstante, sí que parece que los resultados mejoran cuando se descartan más datos del inicio de cada clase o cuando se utilizan 500 muestras para el cálculo de la FFT en lugar de 250. Esto sólo ha podido verificarse cuando se analizaba los datos de uno o varios periodos utilizando validación cruzada, como se muestra en la tabla 5.13 para el caso del conjunto de datos recogidos durante la primera sesión de adquisición realizada por David.

Estos resultados indican que los datos se comportan como si se estuviesen clasificando mediante un clasificador aleatorio. No es posible abstraer características comunes de los datos de una misma clase que permitan, dado un patrón, discernir si pertenece a una clase u otra. Hay que tener en cuenta que en el ámbito de los sistemas BCIs no siempre se pueden clasificar los datos adquiridos, ya que éstos dependen de la capacidad del usuario. Existe el fenómeno denominado *BCI-Illiteracy* (ya mencionado en la sección 2.3) del que todavía no se sabe demasiado pero que se verifica en los diferentes experimentos realizados con sistemas BCIs: hay un porcentaje de la población (se estima que un 20 %) que es incapaz de modular sus señales cerebrales de forma que sean clasificables por un sistema BCI. Puede que los sujetos que han participado como voluntarios en este proyecto de fin de carrera pertenezcan a ese grupo de personas. En el proyecto realizado por Javier Asensio [2] participaron dos voluntarios que realizaron varias sesiones de adquisición. Con uno de ellos sí que consiguieron resultados aceptables (en torno al 70 % de aciertos al clasificar dos clases) y el sujeto llegó a controlar el cursor por la pantalla con bastante éxito mediante la aplicación incluida en el programa MindReader. Con el otro sujeto participante los resultados fueron similares a los obtenidos en el desarrollo de este proyecto de fin de carrera.

Capítulo 6

Conclusiones y trabajos futuros

En este capítulo se analizan los objetivos alcanzados, prestando especial atención a si éstos satisfacen los objetivos iniciales con los que surgió el proyecto, y se recogen las posibles líneas futuras de ampliación y mejora del sistema.

6.1. Conclusiones

En la sección 1.2 se especificaron los objetivos planteados al inicio del proyecto.

Respecto al primer objetivo, la adquisición de datos electroencefálicos mediante el sistema BCI existente, podemos decir que sólo se ha satisfecho parcialmente. Se han obtenido datos electroencefálicos de varias personas, con una de ellas se han realizado varias sesiones de adquisición, pero en ningún caso se ha conseguido obtener un clasificador que pueda utilizarse satisfactoriamente en el sistema BCI para utilizar procesos mentales como sentencias de control de dispositivos (como podría ser el control de la actividad de un cursor).

Al analizar los datos parece que los voluntarios que han participado pertenecen al grupo de personas que no son capaces de modular sus procesos mentales para que éstos sean clasificados de forma automática. Como ya se comentó en la sección 5.3, "Análisis de los datos adquiridos", con los datos adquiridos se obtienen clasificadores que proporcionan resultados de aciertos similares a los de un clasificador aleatorio. Los resultados sólo mejoran un poco en el caso del análisis de la primera sesión de adquisición realizada por David, pero aún así éstos no son suficientemente buenos.

La baja o nula calidad de los clasificadores obtenidos ha impedido que pudiesen realizarse sesiones que implicasen su utilización, como son las se-

siones con retroalimentación o sesiones de clasificación en las que controlar la actividad de un cursor a través del sistema BCI.

Además, las primeras sesiones de adquisición se realizaron sin haber hecho un análisis exhaustivo de la herramienta. Esto provocó que, por desconocimiento, se cometieran una serie de errores a la hora de especificar los parámetros a emplear. Estos errores hubo que subsanarlos a posteriori modificando a mano los ficheros de datos obtenidos. Este hecho nos recuerda la importancia de entender bien las herramientas que se van a utilizar antes de hacerlo, lo que en un principio parece una pérdida de tiempo puede ahorrar muchos problemas en el futuro y a medio y largo plazo reducir los recursos temporales requeridos para realizar determinada tarea compleja.

El hecho de no haber conseguido finalmente buenos resultados con las sesiones de adquisición ha sido bastante desalentador. En primer lugar no se sabía si los malos resultados serían debidos a un error en los parámetros, o en el procesado que realiza la aplicación. Después de revisar a fondo el código fuente de la aplicación y subsanar los errores cometidos en los parámetros, los resultados seguían siendo muy malos. Se realizaron a continuación distintos procesados y análisis, como se detalla en el apartado 5.3, pero no se consiguió obtener buenos clasificadores. Finalmente hubo que concluir que el problema estaba en los propios datos recogidos, y no en el procesamiento o en los análisis realizados.

El segundo objetivo principal del proyecto era mejorar la herramienta principal del sistema BCI, la aplicación MindReader. Podemos concluir que este objetivo ha sido alcanzado con bastante éxito. Se han añadido las funcionalidades que se habían planteado al inicio del proyecto como prioritarias: la posibilidad de aplicar un filtro a los datos, la posibilidad de reentrenar redes de neuronas, y la posibilidad de simular sesiones, recalculando en cada simulación la estimación del PSD.

El hecho de que el filtro pueda seleccionarse de forma dinámica es un aporte importante para la aplicación, ya que hasta el momento si se deseaba realizar un procesamiento a los datos distinto al establecido, había que tocar el código fuente de la aplicación y recompilarlo. Ahora se permite modificar los datos del dominio del tiempo multiplicándolos por una matriz que se selecciona de forma muy cómoda indicando, a través de la interfaz gráfica, el fichero que contiene su especificación.

La simulación de sesiones permite dos cosas importantes. En primer lugar, se puede utilizar la aplicación MindReader sin necesidad del hardware y software de *Brain Vision*. En segundo lugar, se puede recalcular el PSD con diferentes parámetros tantas veces como se desee desde la propia aplicación. Esto no era posible antes salvo que se hiciera a través de una aplicación externa, como, por ejemplo, empleando un script para Octave que calculase

el PSD y lo volcase a un fichero.

Además se han integrado las funcionalidades del programa nnt en la aplicación MindReader, de forma que por una parte se conservan las funcionalidades originales de la aplicación nnt (entrenamiento de redes neuronales a partir de ficheros con los datos de la estimación del PSD, y generación de archivos `.arff` a partir de dichos datos) y por otra se añaden otras nuevas que surgen de la integración de los dos programas (se automatiza el proceso de generación de clasificadores al finalizar una sesión de adquisición o al término de una simulación).

El programa MindReader dispone ahora de funcionalidades relacionadas con las que realizaba la aplicación nnt pero que no estaban implementadas en dicha herramienta, como es la posibilidad del entrenamiento incremental de clasificadores de tipo redes neuronales artificiales. Se permite reentrenar una red de neuronas tomando como pesos de partida los de una red ya existente.

Se han detectado otras carencias de la herramienta, y algunas de ellas se han solventado, como es la necesidad de disponer de un instalador para la aplicación. No obstante, todavía quedan bastantes aspectos que pueden ser mejorados. En el apartado siguiente, 6.2 "Trabajos Futuros", se comentan algunos de ellos.

6.2. Trabajos futuros

En el apartado 4.2 "Aspectos a mejorar de la herramienta" se enumeraron una serie de carencias que se habían detectado en la herramienta MindReader y algunas de ellas han quedado pendientes de solucionar. En esta sección se van a recordar estos aspectos y se van a comentar otras vías de continuación de este proyecto que resultan interesantes.

Sería interesante implementar más clasificadores y más procesadores de datos. Actualmente el único tipo de clasificador que se puede utilizar en la aplicación MindReader son las redes de neuronas de tipo perceptrón multicapa, pero existen otros que podrían dar también buenos resultados (como las redes de neuronas de base radial o el algoritmo SMO). No obstante, los resultados positivos que no puedan obtenerse con una red de neuronas MLP tampoco se obtendrán al aplicar otro clasificador, ya que está comprobado que las redes de neuronas MLP funcionan bien en el dominio de los sistemas BCIs, aunque no son los únicos clasificadores que así lo hacen.

Otra funcionalidad que sería muy interesante incluir es que se permitiese el entrenamiento incremental de clasificadores durante las sesiones de adquisición con retroalimentación. Esto implicaría que, cada cierto número de muestras recogidas en tiempo real, automáticamente se reentrenase el

clasificador con dichos datos, generando un nuevo clasificador. La sesión de adquisición con retroalimentación continuaría utilizando el nuevo clasificador. Este proceso se repetiría varias veces durante la sesión.

También sería interesante que los procesadores pudiesen seleccionarse de forma dinámica, indicando, en tiempo de ejecución, qué procesadores se quiere aplicar a los datos, a través de una interfaz gráfica que permita seleccionarlos e indicar el orden de ejecución.

Sería muy recomendable modificar la aplicación para que se pueda especificar explícitamente qué datos se utilizan para el conjunto de entrenamiento y cuales para el conjunto de validación, de forma que podrían seleccionarse varios ficheros de datos pertenecientes a una sesión de adquisición para su uso como conjunto de entrenamiento, y seleccionar otros, compatibles con los anteriores, para utilizarlos como conjunto de validación. Este tipo de análisis sólo puede realizarse actualmente mediante el uso de aplicaciones externas, como WEKA.

Respecto a los parámetros que hay que introducir en la aplicación, como se ha comentado varias veces a lo largo de esta memoria, hay algunos (*patrones a descartar*, *patrones generados por clase*) cuyos valores están muy ligados a los valores de otros parámetros. Una mejora de la herramienta que facilitaría la vida al usuario sería que dichos parámetros fuesen calculados de forma automática una vez que se hayan fijado el resto. Además el descarte de los primeros patrones de cada clase, para los cuales aún no se tienen los datos necesarios para calcular el PSD al no disponer de suficientes datos anteriores, debería realizarse automáticamente.

Para mejorar la usabilidad del sistema, deberían revisarse todas las interfaces de ventana de la aplicación, y unificarlas para que utilicen un lenguaje similar en sus títulos y en los rótulos de los controles. Estaría muy bien añadir una ayuda contextual emergente cada vez que se pase el cursor por encima de un control y un menú de ayuda donde el usuario pueda consultar el uso de la aplicación. Deberían realizarse controles de completitud y formato de todos los parámetros y ficheros a cargar e introducirse cuadros de progreso mientras el sistema está ocupado, para que el usuario no se impaciente en las operaciones que llevan más tiempo y no tenga la sensación de que el sistema no responde.

Aunque se ha intentado depurar el código fuente original, todavía faltaría por detectar y eliminar ciertos fallos de memoria y pérdidas. También debería mejorar el rendimiento del procesamiento de datos (se sigue utilizando un solapamiento de un patrón para el cálculo del PSD, lo que supone realizar esta operación algo costosa 500 veces por segundo en lugar de las 16 que serían necesarias), y optimizar las lecturas y escrituras.

Por último, comentar que también se echa en falta un estudio sobre los

límites de procesado de la aplicación, donde se averigüe, por ejemplo, cúantas instancias pueden clasificarse en las diferentes sesiones de adquisición con retroalimentación sin comprometer los requerimientos de tiempo real de la aplicación.

Apéndice A

Guía de instalación

Se ha generado un instalador que permite instalar y desinstalar la aplicación MindReader de forma cómoda en entornos Windows. Se ha verificado su correcto funcionamiento en máquinas con sistema operativo MS Windows Vista y MS Windows XP. En esta sección se explican los pasos a seguir para instalar o desinstalar el programa. La instalación/desinstalación se realiza a través de una serie de ventanas por las que se navegará empleando los botones "atrás" y "siguiente". En cualquier momento de la instalación o desinstalación se puede pulsar el botón "cancelar" y se abortará el proceso que se esté realizando.

A.1. Instalación

Se debe ejecutar el instalador `Setup.exe` o `SetupMindReader2.msi`, se mostrará durante unos segundos la pantalla de carga del instalador (ver figuraA.1) y aparecerá a continuación la ventana de inicio del instalador (ver figura A.2), se debe pulsar el botón *siguiente*.

Figura A.1: Carga del instalador

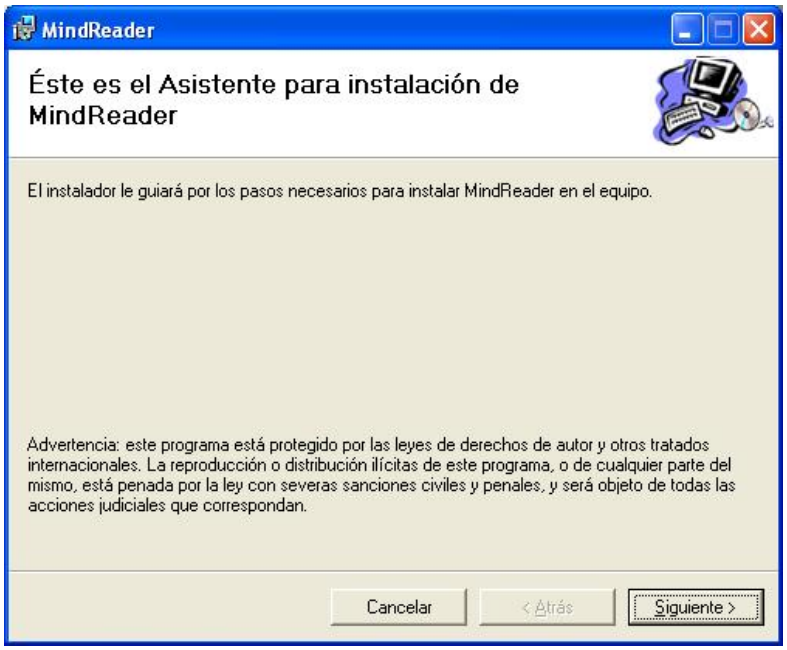

Figura A.2: Ventana de bienvenida del instalador

En la siguiente pantalla se puede elegir la carpeta de instalación y si la instalación se realiza para todos los usuarios o sólo para el que ha iniciado la sesión (ver figura A.3). Se permite consultar el espacio disponible en disco. A la hora de elegir la ubicación de la instalación se debe tener en cuenta que la instalación mínima ocupa 13.3MB y la instalación completa 200 MB libres.

Al pulsar "siguiente" aparece la ventana que nos permite elegir qué contenido opcional instalar (ver figura A.4). Según qué opciones se marquen se incluirán en el directorio de instalación diversos ficheros de ejemplo utilizados por la aplicación.

Por defecto viene marcada opción de "datos de configuración" que supone 617 Bytes adicionales con un ejemplo de fichero de configuración que se puede utilizar para cargar parámetros desde cualquiera de las ventanas de preferencias de la aplicación.

Si se selecciona la opción "datos de ejemplo _time.raw, _psd.raw y _patterns.mrp" se incluirá un ejemplo de archivo de cada uno de éstos, que contienen los datos obtenidos durante una sesión de adquisición. Hay que tener en cuenta que estos archivos ocupan 185 MB.

Si se selecciona la opción "archivos de filtros" se incluirán tres ejemplos de archivos de filtros que ocupan 1.44 KB.

Al pulsar siguiente aparecerá la ventana de confirmación de la instalación (ver figura A.5). Al pulsar el botón siguiente comenzará la instalación (ver

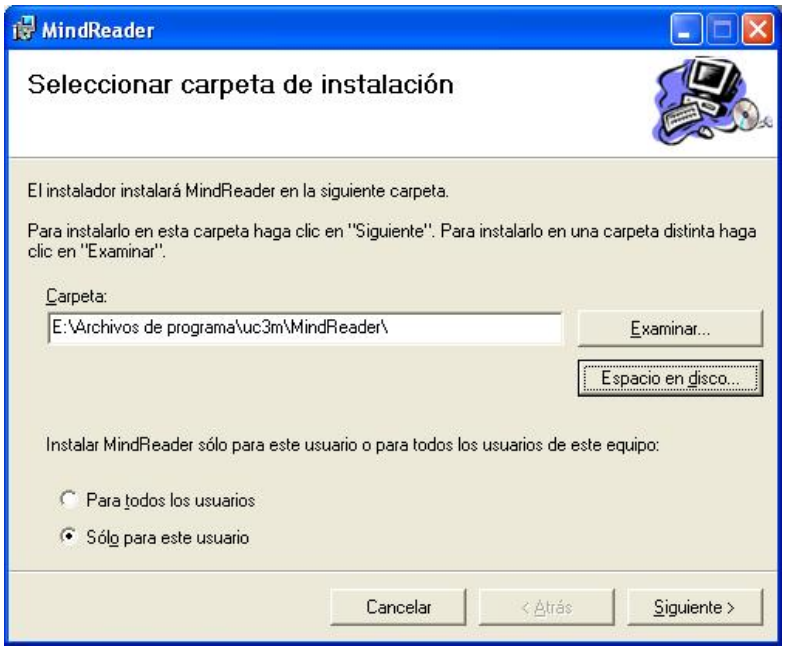

Figura A.3: Selección de carpeta de instalación

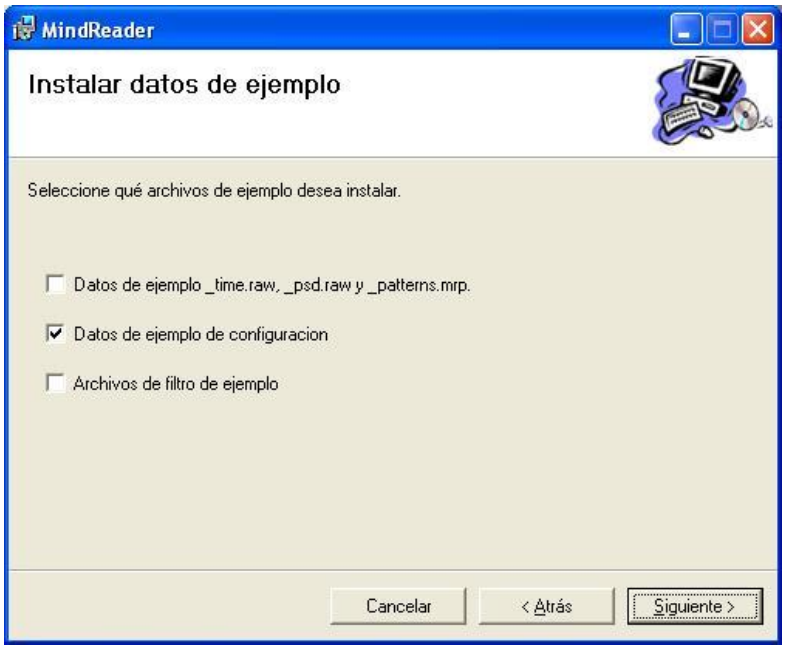

Figura A.4: Elementos adicionales: datos de ejemplo

figura A.6) y concluirá mostrando la ventana que se muestra en la figura

A.7. Al pulsar el botón “cerrar” de dicha ventana el proceso de instalación habrá concluido.

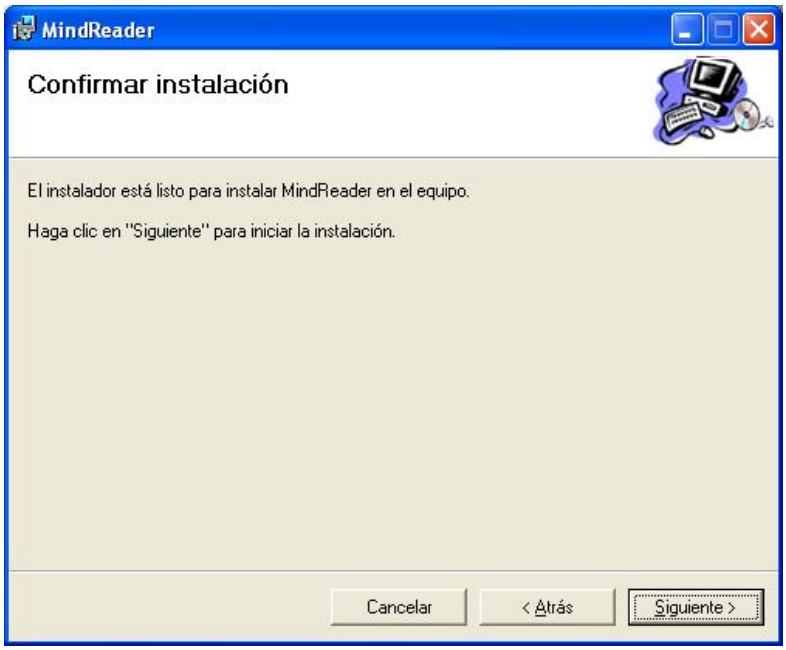

Figura A.5: Confirmación de la instalación

Si en algún momento del proceso de instalación se pulsa el botón “cancelar” de alguna de las ventanas aparecerá la ventana que se muestra en la figura A.8. Se debe pulsar el botón “cerrar”.

A.2. Desinstalación de la aplicación

Se puede desinstalar la aplicación utilizando el mismo archivo SetupMindReader2.msi que se utilizó para instalarla. Cuando se ejecuta dicho archivo en un equipo donde MindReader está instalado, aparecerá la ventana que se muestra en la figura A.9. Se debe seleccionar la opción “quitar MindReader” y pulsar siguiente, entonces aparecerá la ventana de la figura A.10 que muestra el progreso de la desinstalación. Se indica que la desinstalación ha concluido mediante la ventana que se muestra en la figura A.11, se debe pulsar el botón “cerrar”.

Después de desinstalar el programa, puede que haya que eliminar manualmente las carpetas donde estaba instalado el programa, que estarán vacías (si se dejó la opción predeterminada, esto será c:\Archivos de Programa\uc3m\MindReader.)

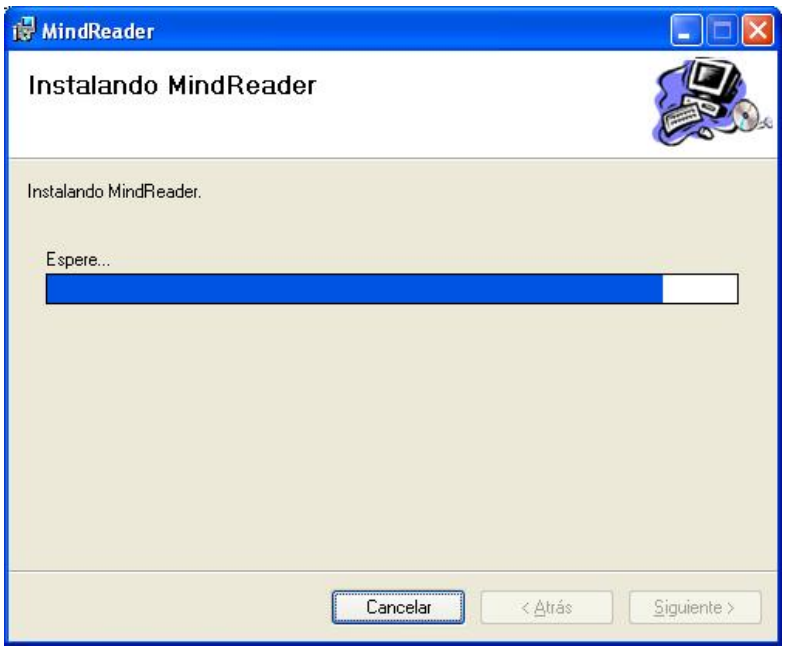

Figura A.6: Realizando instalación

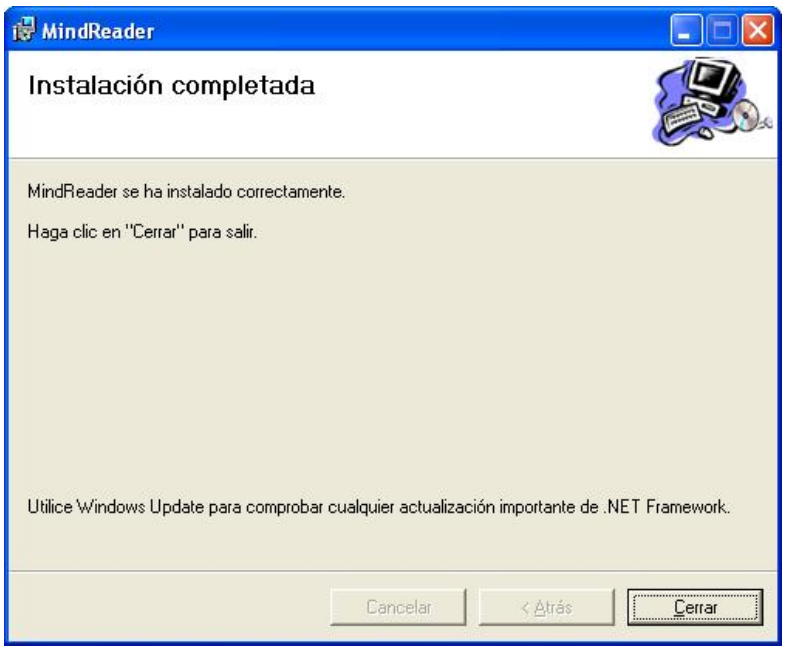

Figura A.7: Instalación concluida

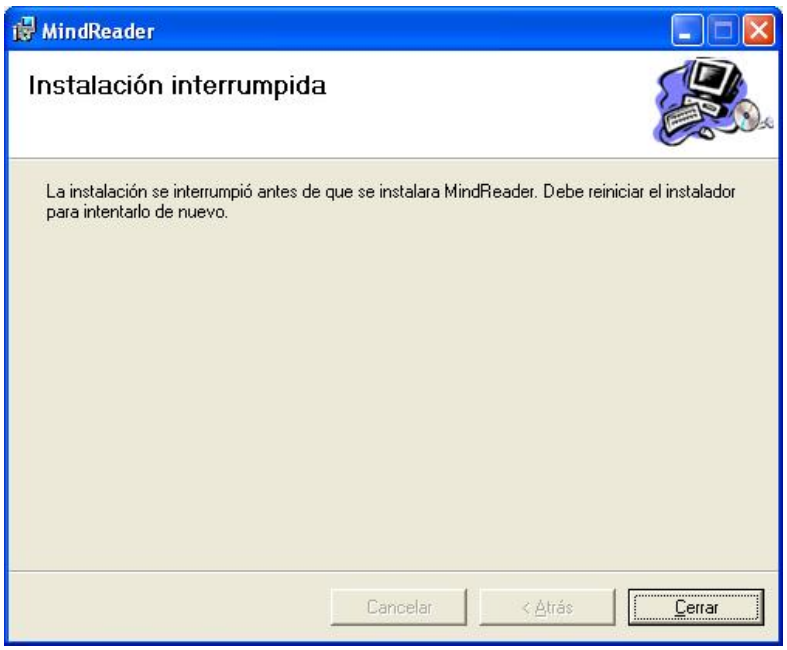

Figura A.8: Instalación interrumpida

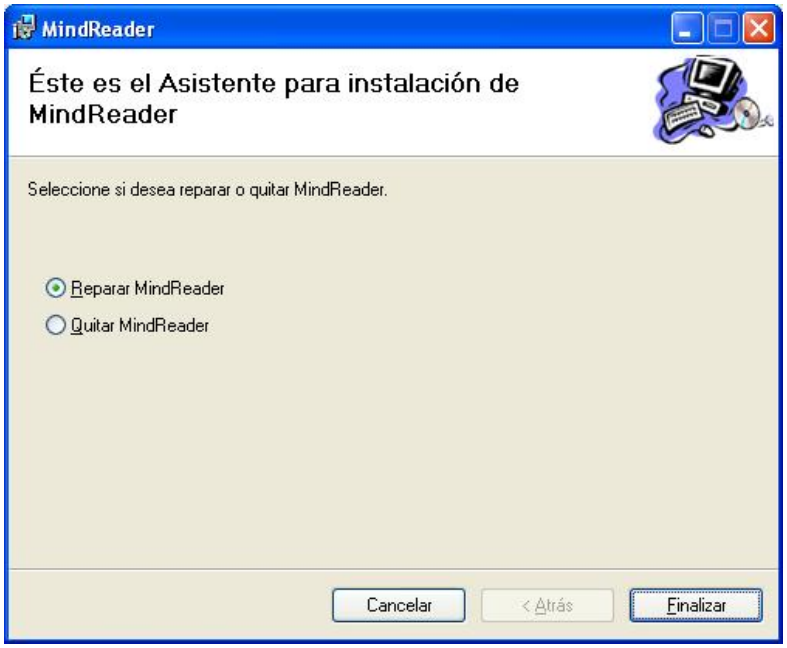

Figura A.9: Desinstalar

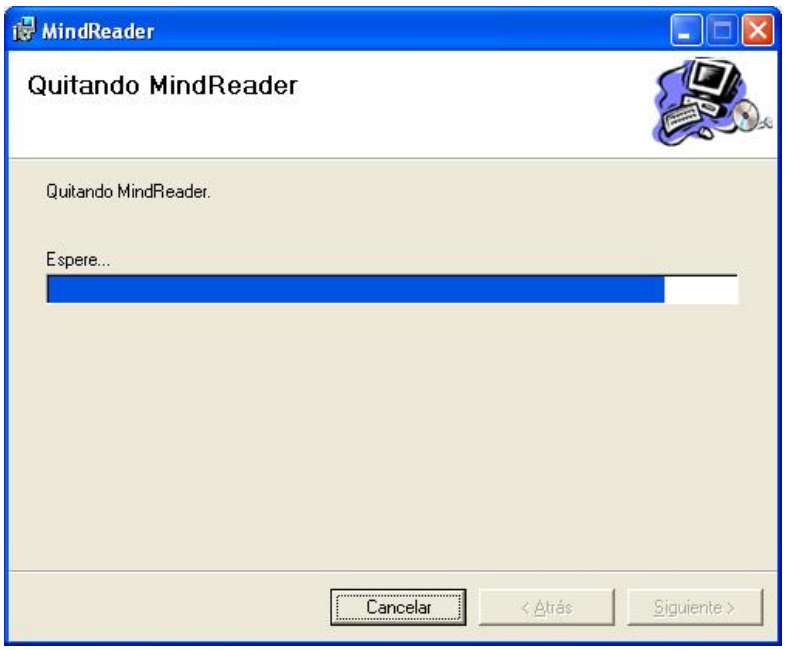

Figura A.10: Desinstalando

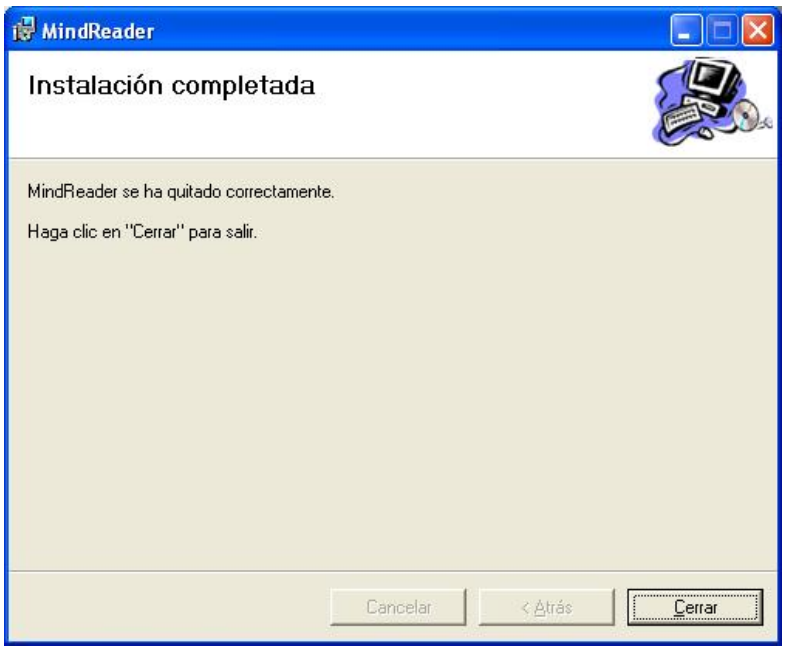

Figura A.11: Desinstalación finalizada

Apéndice B

Presupuesto

En este anexo presentamos el presupuesto asociado al desarrollo del presente proyecto. Este presupuesto es ficticio, algunos conceptos cuyo coste se detalla en realidad no suponen gasto económico alguno, pero se detallan los gastos que supondría realizar este proyecto en un entorno empresarial (sin trabajo voluntario ni beneficios por pertenecer a la comunidad universitaria). En estos casos figura una nota al respecto en el pie de página.

En primer lugar, se detallan los costes de personal en que se va a incurrir y, a continuación, se muestran los costes derivados del material a emplear. La adición de cada uno de estos costes junto con el coste asociado a riesgos/gastos indirectos y el beneficio esperado constituirá el presupuesto final. No se han contemplado los gastos derivados del uso de las infraestructuras propias de la empresa/entidad donde se realice el desarrollo del proyecto (alquiler/compra de inmueble, gastos de gas, electricidad, agua, etc.)

El presupuesto final que refleja el coste total con el IVA incluido que supondrá el desarrollo del proyecto asciende a **44.193,68 euros (cuarenta y cuatro mil ciento noventa y tres euros con sesenta y ocho céntimos de euro)**.

B.1. Costes de personal

Los costes de personal incluyen los honorarios del Ingeniero Informático encargado del desarrollo del proyecto y de los dos expertos en Inteligencia Artificial y sistemas BCIs que le asesoran, así como la compensación económica asignada a los voluntarios.

Se toma como valor orientativo que los honorarios de un **Ingeniero Informático** especializado en Inteligencia Artificial sin experiencia es de **20 euros por hora**. El proyecto ha sido realizado entre los meses de Octubre

del 2008 a Mayo del 2010, durante los cuales el ritmo de trabajo no ha sido constante, se estima que se han invertido **1.100 horas** de trabajo durante dicho periodo. Por tanto, el coste asociado a dicho concepto asciende a **22.000 euros** (veintidos mil euros).

Se estima que los honorarios de los **expertos** que asesoran al ingeniero desarrollador ascienden a **35 euros por hora**. Se realizarán varias consultas puntuales a los mismos y se calcula que en total se requerirán **100 horas** de trabajo por parte de los expertos. Por tanto, el coste asociado a este concepto asciende a **3.500 euros** (tres mil quinientos euros).

Se incluyen además los costes de la compensación económica percibida por los **voluntarios** que participarán en el proyecto en las sesiones de adquisición de datos, estimada en **50 euros por sesión**. Se realizan cinco sesiones de adquisición de datos, por lo que el coste asociado a este concepto es de **250 euros** (doscientos cincuenta euros).

Por tanto, el coste total debido a **gastos en personal** asciende a **25.750 euros** (veinticinco mil setecientos cincuenta euros). La tabla B.1 recoge este resultado.[1]

Concepto	**Trabajo realizado**	**Honorarios**	**Importe**
Ingeniero Informático	1.100 horas	20 €/hora	22.000€
Expertos en IA y BCI	100 horas	35 €/hora	3.500€
Sujetos experimentación	5 sesiones	50 €/sesión	250€
Total gastos personal			**25.750€**

Tabla B.1: Costes de personal

B.2. Costes de material

Los materiales a emplear durante la realización del proyecto son los siguientes:

- Un ordenador personal de sobremesa con procesador Intel Core Duo a 2.33 GHz, con 1GB de memoria principal. Esta máquina está valorada aproximadamente en 600 euros. Considerando como periodo de amortización del aparato 3 años, y considerando que el uso dado a mismo

[1] En realidad ninguno de los participantes (ni los tutores del proyecto que han asumido el rol de expertos, ni la autora del PFC que ha asumido el rol de ingeniera desarrolladora, ni los tres voluntarios que participaron en las sesiones de adquisición) han recibido compensación económica alguna asociada al proyecto.

durante el proyecto no es exclusivo, es dificil calcular el coste asociado al proyecto derivado de la utilización de dicha máquina, pero puede estimarse en **200 euros** (doscientos euros).

- Un ordenador portátil HP compact 6720s con procesador Intel Celeron a 1.73 GHz, con 1GB de memoria principal. Esta máquina está valorada aproximadamente en 700 euros. Considerando como periodo de amortización del aparato 3 años, y considerando que el uso dado a mismo durante el proyecto no es exclusivo, puede estimarse su coste en **200 euros** (doscientos euros).

- Equipo completo para la recogida de datos encefalográficos, compuesto por el casco encefalógrafo modular Easy-Cap con electrodos de anilla pasivos, un amplificador V-amp de 8 canales de BrainProducts y el software BrainVision Recorder V-Amp edition de BrainProducts. Incluye materiales fungibles en cantidades superiores a las requeridas para las sesiones de adquisición realizadas (torundas, alcohol y gel electrolito abrasivo). Estos componentes están valorados en **12000 euros**. Considerando que el periodo de amortización del equipo es de 6 años, y que ha sido utilizado en exclusiva para este proyecto durante 20 meses, se puede estimar el coste asociado al uso de este equipamiento en **3.333,33 euros** (tres mil trescientos treinta y tres euros con treinta y tres céntimos de euro). [2]

- Las máquinas tienen instalado un sistema operativo Microsoft Windows XP, Microsoft Windows Vista, cuyas licencias están valoradas en **57 euros** (cincuenta y siete euros) y **65 euros** (sesenta y cinco euros) respectivamente. Además se ha utilizado el IDE Microsoft Visual Studio 2005, valorado en **53 euros** (cincuenta y tres euros) y DevPartner for Visual C++ BoundsChecker Suite - Visual Studio Edition cuya licencia está valorada en **1.250 euros** (mil doscientos cincuenta euros). [3]

- Conexión a Internet durante la realización del proyecto. Es necesaria para conseguir documentación para el proyecto y como medio de co-

[2]Durante el proyecto se podría haber compartido el uso de este equipo, y esto habría abaratado el coste asociado a este concepto.

[3]Precios obtenidos de http://www.componentsource.com/products/, http://latam.preciomania.com/ y de http://www.ciao.es/. En el proyecto no se ha producido ningún gasto asociado al uso de la herramienta BoundChecker ya que se ha utilizado una versión de evaluación. Tampoco se ha producido ningún gasto asociado al uso del software de Microsoft, ya que se han utilizado las licencias gratuitas que se proporcina a estudiantes y docentes de la UC3M gracias al programa al que está adscrita la universidad, "MSDN Academic Alliance".

municación entre los expertos y el desarrollador, para concertar las sesiones de adquisición con los voluntarios... Está valorada aproximadamente en 42 euros al mes, que multiplicado por 20 meses, supone un coste de **840 euros** (ochocientos cuarenta euros).

- El resto del software utilizado durante el desarrollo del proyecto (WEKA, Octave, LaTeX,...) es gratuito y de libre distribución.

Teniendo en cuenta todos estos elementos, los costes de material ascienden a **5.998,33 euros** (cinco mil novecientos noventa y ocho euros con treinta y tres céntimos de euro), se detallan en la tabla B.2.

Concepto	**Importe**
Ordenador sobremesa	200€
Ordenador portátil	200€
Equipo recogida datos EEG	3.333,33€
Licencia MS Windows vista	65€
Licencia MS Windows XP	57€
Licencia MS Visual Studio 2005	53€
Licencia DevPartner BoundChecker	1.250€
Conexión a Internet	840€
TOTAL	**5.998,33€**

Tabla B.2: Presupuesto: Costes de material

B.3. Presupuesto total

A continuación se calcula el coste total del proyecto, teniendo en cuenta el coste de los recursos humanos y materiales necesarios para la realización del mismo, así como los riesgos y gastos indirectos y los beneficios asociados al mismo. Se va considerar el riesgo máximo y gastos indirectos del 15 %, mientras que el beneficio será del 5 %. Como se observa en la tabla B.3, el total asciende a **44.193,68 euros (cuarenta y cuatro mil ciento noventa y tres euros con sesenta y ocho céntimos de euro)**.

Concepto	**Importe**
Costes de personal	25.750€
Costes de material	5.998,33€
Riesgos y costes indirectos (15 %)	4.762,25€
Beneficio (5 %)	1.587,42€
TOTAL	38.098,00€
IVA (16 %)	6.095,68€
TOTAL con IVA	**44.193,68€**

Tabla B.3: Coste total

Apéndice C

Contenido de los DVDs

Los resultados obtenidos durante el desarrollo de este PFC se incluyen en dos DVDs, `DVD_1de2` y `DVD_2de2`, cuyo contenido se especifica en este anexo.

C.1. Contenido del DVD `DVD_1de2`

En `DVD1/2` se incluyen los datos EEG obtenidos durante las sesiones de adquisición de datos, así como los resultados de simular las sesiones de adquisición para recalcular la estimación del PSD. Por falta de espacio, el contenido relativo a la sesión de adquisición de Pablo se ubica en el DVD `DVD_2de2`. El contenido por directorios de este DVD se indica a continuación:

- datosEEG_1de2:

  El directorio con los datos obtenidos durante las ejecuciones del programa MindReader.

- datosEEG_1de2/X:

  Datos de la sesión de adquisición X

- datosEEG_1de2/X/datosRAWxxxx.rar:

  Archivo comprimido que contiene los datos generados durante la sesión de adquisición xxxx en formato del programa MindReader. Contiene tanto los ficheros con los datos en el dominio del tiempo como los de los datos en el dominio de la frecuencia.

- datosEEG_1de2/X/A[N]xxxxMRP.rar:

  Archivo comprimido con los ficheros `*_patterns.mrp` con los parámetros utilizados para generar los datos del análisis [N].

- datosEEG_1de2/X/A4datosPSDxxxx.rar:

  Archivo comprimido con los ficheros que contienen los datos del PSD de la sesión xxxx calculados según los parámetros del *Análisis 4* mediante simulación.

C.2. Contenido del DVD `DVD_2de2`

En este DVD se encuentran la memoria del PFC, el código de la aplicación MindReader, su instalador, los scripts de octave así como la aplicación nnt y los resultados de los análisis realizados a los datos adquiridos (archivos arff, salida del programa weka, informe de errores del entrenamiento de redes neuronales...). También se encuentran los datos de la señal en crudo de la sesión de adquisición de Pablo, que no se incluyeron en el `DVD_2de2` por falta de espacio.

- memoriaPFC.pdf

  Memoria del proyecto de fin de carrera "Mejora de la herramienta MindReader. Adquisición y análisis de señales EEG.", realizado por María Teresa Luque Ibañes. Es decir, el documento donde se incluye este anexo.

- datosEEG/2de2/datosEEGpablo

  Datos EEG obtenidos durante la sesión de adquisición de datos de Pablo, así como el resultado de simular dicha sesión de adquisición para recalcular la estimación del PSD.

- mindreader_2.0:

  Contiene el código fuente de la aplicación MindReader, con las nuevas funcionalidades y mejoras incorporadas.

- mindreader_2.0/src:

  Codigo fuente de la aplicacion, parte portable no dependiente de las bibliotecas de microsoft.

- mindreader_2.0/vs/mindreader:

  Codigo fuente no portable (Interfaz de usuario) y ficheros para MS Visual Studio.

- mindreader_2.0/lib:

  Bibliotecas de terceros utilizadas en el desarrollo del programa.

- util/octave:

  Scripts de octave para la generacion de arffs partiendo de los ficheros [prefijo]_time.raw, sin desordenar los patrones. El script `time2arff.m` sirve para periodos de adquisición en los que se utilicen dos clases, y `time2arff3.m` si se han utilizado tres clases. En el archivo `ej_time2arff_pablo_s1.m` se muestra un ejemplo de cómo generar archivos con el PSD (a partir de los datos de la primera sesión de adquisición de Pablo) utilizando estos scripts.

- util/nnt

  Ejecutable del programa nnt junto con la librería requerida.

- SetupMR/SetupMindReader2:

  Ficheros para el MS Visual Studio del proyecto de instalación de la aplicación MindReader.

- SetupMR/SetupMindReader2/Release:

  Ficheros de instalación de la aplicación en MS Windows (se puede ejecutar tanto `setup.exe` como `SetupMindReader2.msi` para proceder a la instalación del programa).

- AnalisisDatos:

  Ficheros obtenidos en los análisis de datos.

- AnalisisDatos/X:

  Ficheros obtenidos al analizar la sesión de adquisición X.

- AnalisisDatos/X/A[N]xxxx

  Ficheros obtenidos al analizar mediante los parámetros del *Análisis [N]* los datos de la sesión xxxx.

  Contiene los archivos arff para ser procesados con el programa WEKA, obtenidos a partir de los datos de la sesión xxxx, según los parámetros del `Análisis [N]`. Si los ficheros han sido generados mediante Octave, llevarán el prefijo `_OCT`.

  Además contiene los ficheros de información de errores obtenidos durante el entrenamiento de redes de neuronas con la aplicación MindReader o nnt (*_errors.txt) y la salida generada por el programa WEKA al generar clasificadores de tipo SMO o MLP. En algunos casos se incluyen también los clasificadores generados.

Apéndice D

Acrónimos y Abreviaturas

BBCI (*Berlin Brain-Computer Interface*).

BCI (*Brain-Computer Interface*) Interfaz Cerebro-Ordenador.

BOLD response *Blood Oxygen Level Dependent Response*). Variaciones del nivel de oxígeno en sangre.

CPU (*Central Processing Unit*). Unidad Central de Proceso.

CSP (*Common Spatial Patterns*) Patrones Espaciales Comunes.

DFT (*Discrete Fourier Transform*) Transformada de Fourier Discreta.

DVD (*Digital Versatil Disc*)Disco Versátil Digital.

ECoG Electrocorticograma.

EEG Electroencefalógrafo o electroencefalograma.

ERD/ERS (*Event Related Desynchronization/Event Related Synchronization*) Desincronización/Sincronización asociada a eventos.

ERPs (*Event-Related Potentials*). Potenciales relacionados con eventos o potenciales evocados.

EVANNAI (*EVolutionary Algorithms and Neural Networks and Artificial Intelliegence*). Grupo de Computación Evolutiva y Redes Neuronales de la UC3M.

FANN (*Fast Artificial Neural Network*). Redes neuronales artificiales rápidas.

FFT (*Fast Fourier Transform*). Transformada rápida de Fourier.

FIFO (*First In, First Out*). Primero en entrar, Primero en salir.

fMRI (*functional Magnetic Resonance Imaging*). Imagen por resonancia magnética funcional.

GB Gigabytes.

GHz Gigahercios.

Hz Hercios.

ICA (*Independent Component Analysis*). Análisis de Componentes Independientes.

IDE (*Integrated Development Environment*. Entorno de Desarrollo Integrado.

IVA Impuesto de Valor Añadido.

MEG Magnetoencefalógrafo o magnetoencefalograma.

MFC Microsoft Foundation Classes.

MLP (*MultiLayer Perceptron*). Perceptrón Multicapa.

NIRS (*Near Infrared Spectroscopy*)Espectrografía del infrarrojo cercano o topografía óptica.

ms milisegundos.

PCA (*Principal Component Analysis*) Análisis de componente principal.

PCLs Potenciales Corticales Lentos.

PCRMs Potenciales Cerebrales Relacionados con el Movimiento.

PSD (*Power Spectral Density*). Densidad espectral de potencia.

PFC Proyecto de Fin de Carrera.

SCP (*Slow Cortical Potentials*). Potenciales corticales lentos (PCLs).

SMO (*Secuential Minimal Optimization*). Algoritmo de entrenamiento y optimización para máquinas de soporte vectorial, que descompone el problema de optimización en tareas mucho más pequeñas reduciendo el tamaño de las operaciones matriciales.

SMR (*Sensorimotor Rythm*). Ritmo sensoriomotor.

SSVEP (*Steady-state Visual Evoked Potential*). Potencial evocado visual continuo.

TCP/IP (*Transmision Control Protocol over Internet Protocol*) Protocolo de control de transmisión sobre protocolo de Internet.

TFT (*Thin Film Transistor*). Transistor de película fina.

UC3M Universidad Carlos III de Madrid.

USB (*Universal Serial Bus*). Bus universal en serie.

Bibliografía

[1] Página oficial de EVANNAI http://www.evannai.uc3m.es/

[2] Javier Asensio, PFC "Diseño de un interfaz cerebro-máquina". Ingeniería Superior en Informática. Septiembre 2008. Universidad Carlos III

[3] Artículo sobre el implante clocear http://www.susmedicos.com/articulos_otologia_coclear1.htm

[4] Página de "Retina Implant AG" http://www.retina-implant.de/en/about/

[5] Guido Dornhege, José del R. Millán, Thilo Hinterberger, Dennis J. McFarland and Klaus-Robert Müller; Toward Brain-Computer Interfacing;2007 MIT press; ISBN 978-0-26204244-4.

[6] Medciclopedia: Sistema Nervioso Central http://www.iqb.es/neurologia/a002.htm

[7] Martin,J.H. Neuroanatomía, 2.ed..Prentice Hall. Madrid (España).1998.

[8] Young, P.A., Young P. H.. Neuroanatomía Humana, 4a ed. Masson. Barcelona (España). 2004.

[9] Anatomía del encéfalo. Universidad de Virgina.http://www.hsc.virginia.edu/uvahealth/peds_neuro_sp/anatomy.cfm

[10] Constantina Álvarez Peña. Apuntes "Fundamentos de Bioelectrónica". Universidad de Oviedo. http://www.ate.uniovi.es/14005/documentos/clases%20pdf/encefalo.pdf

[11] Rains, G.D. Principles of Human Neurpsychology. McGraw-Hill. New York (EEUU) . 2002.

[12] Kolb, B., Whishaw, I.Q. Neuropsicología Humana, 5a ed. Médica Panamericana. Madrid (España). 2006.

[13] Página oficial del BBCI http://www.bbci.de/

[14] Wikipedia: 10-20 System http://en.wikipedia.org/wiki/10-20_system_%28EEG%29

[15] Birbaumer, N., Elbert, T., Lutzenberger, W., y Rockstroh, B. Biofeedback of slow cortical potentials Effects on signal detection and reaction time. Proceedings of the Biofeedback Society of America, 1979.

[16] wikipedia: "P300"http://209.85.229.132/search?q=cache:r5K0FSjfaGsJ:es.wikipedia.org/wiki/P300+wiki+p300&cd=1&hl=es&ct=clnk&gl=es&client=firefox-a

[17] Página web Ïnteracción Humano Máquina"http://209.85.229.132/search?q=cache:_vkoDnu2WfAJ:interfacemindbraincomputer.wetpaint.com/page/2.A.4.-Grupos%2Bde%2Binvestigacion%2Bdel%2Bsistema%2BBCI+SSVEP+potencial+evocado&cd=4&hl=es&ct=clnk&gl=es&client=firefox-a

[18] Wikipedia: Magnetografía. http://es.wikipedia.org/wiki/Magnetoencefalograf%C3%ADa

[19] A. Kübler, F. Nijboer, N. Birbaumer. "Brain-Computer Interfaces for Communication and Motor Control- Perspectives on Clinical Application"

[20] http://www.investigacionyciencia.es/03064497000626/RMN_port%C3%A1til.htm

[21] Adrian M. Owen, Martin R. Coleman, Melanie Boly, Matthew H. Davis, Steven Laureys, John D. Pickard. "Detecting Awareness in the Vegetative State". Science, 8 September 2006: Vol. 313. no. 5792, p. 1402. DOI: 10.1126/science.113019

[22] Wikipedia: PCA filters http://en.wikipedia.org/wiki/Principal_component_analysis

[23] Wikipedia: ICA filters http://en.wikipedia.org/wiki/Independent_component_analysis

[24] Guido Dornhege, Matthias Krauledat, Klaus-Robert Müller and Benjamin Blankertz. "General Signal Processing and Machine Learning Tools for BCI Analysis."

[25] Ricardo Aler, Inés M. Galván and José M. Valls. Evolving Spatial and Frequency Selection Filters for Brain-Computer Interfaces

[26] Power Spectral Density Estimation https://ccrma.stanford.edu/~jos/mdft/Power_Spectral_Density_Estimation.html

[27] Welch's method https://ccrma.stanford.edu/~jos/sasp/Welch_s_Method.html

[28] Welch, P. The use of fast Fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms.IEEE Transactions on Audio and Electroacoustics. Volumen 15. 1967

[29] Wikipedia: Transformada Rápida de Fourier http://es.wikipedia.org/wiki/Transformada_r%C3%A1pida_de_Fourier

[30] Wikipedia: Power Spectral Density http://en.wikipedia.org/wiki/Power_spectral_density

[31] Página oficial de Bionic Ibérica S.A. http://www.bionic.es/

[32] Página oficial de Brain Products GmbH http://www.brainproducts.com/

[33] Página oficial de EASYCAP GmbH http://www.easycap.de/easycap/e/products/products.htm#15

[34] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten (2009); The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1.

[35] Página oficial de WEKA http://www.cs.waikato.ac.nz/~ml/weka/index.html

[36] Página oficial de GNU http://www.gnu.org/copyleft/gpl.html

[37] Página oficial de Octave. http://www.gnu.org/software/octave/about.html

[38] Página oficial de Matlabhttp://www.mathworks.com/products/matlab/

[39] Eric C. Leuthardt, M.D., Gerwin Schalk, Ph.D., Jarod Roland, B.S., Adam Rouse, B.S., and Daniel W. Moran, Ph.D. (July 2009); Evolution of brain-computer interfaces: going beyond classic motor physiology; Neurosurgical Focus, Volumen 27, Number 1.

[40] Wikipedia: Brain Computer Interface http://en.wikipedia.org/wiki/Brain-computer_interface

[41] Página Interacción Humano Máquina http://interfacemindbraincomputer.wetpaint.com/page/2.A.3.-Interface%2FInterfaz+Mente-Cerebro+M%C3%A1quina,+Computadora+u+Ordenador+%28Brain+Computer+Interface%29

[42] Página del "1st International Conference on Neuroprosthethic Devices" http://www.bsrc.nctu.edu.tw/icnd/report.php

[43] Drs. Patricio Sandoval, Patricio Mellado; Síndrome de "Locked-In"; Cuadernos de Neurología; Vol XXIV-2000. http://escuela.med.puc.cl/publ/cuadernos/2000/pub_16_2000.html

[44] Wikipedia: Imagen por resonancia magnética funcional http://es.wikipedia.org/wiki/Imagen_por_resonancia_magn%C3%A9tica_funcional

[45] Wikipedia: Neurotecnología http://es.wikipedia.org/wiki/Neurotecnolog%C3%ADa

[46] Patrón de diseño Composite http://www.codeproject.com/KB/cpp/CompositePatternComponent.aspx

[47] Wikipedia: Autocorrelación Autocorrelaci\penalty\@M\hskip\z@skip\unhbox\voidb@x\bgroup\let\unhbox\voidb@x\setbox\@tempboxa\hbox{o\global\mathchardef\accent@spacefactor\spacefactor}\accent19o\egroup\spacefactor\accent@spacefactor\penalty\@M\hskip\z@skip\setbox\@tempboxa\hbox{o\global\mathchardef\accent@spacefactor\spacefactor}\spacefactor\accent@spacefactornhttp://es.wikipedia.org/wiki/Autocorrelaci%C3%B3n

[48] Azorín JM, Iáñez E, Sabater JM, García NM, Pérez C, Fernández E. Interfaz cerebral no invasiva para control de un sistema domótico por personas discapacitadas. Trauma Fund MAPRE (2009) Vol 20 nº4:249-254.